Improved Hybrid Opponent System for Professional Military Training

Michael Pelosi[*1], Michael Brown[2], Kinza Ahmad[3]

[1]*Professor, East Central University, 1100 E 14th St, Ada, OK 74820, USA*

[2]*Program Director, University of Maryland University College, 1616 McCormick Dr, Upper Marlboro, MD 20774, USA*

[3]*Department of Computer Science, University Institute of Information Technology, PMAS Arid Agriculture University, 46300, Pakistan*

A R T I C L E   I N F O

A B S T R A C T

Described herein is a general-purpose software engineering architecture for autonomous, computer controlled opponent implementation in modern maneuver warfare simulation and training. The implementation has been developed, refined, and tested in the user crucible for several years. The approach represents a hybrid application of various well-known AI techniques, including domain modeling, agent modeling, and object-oriented programming. Inspired by computer chess approaches, the methodology combines this theoretical foundation with a hybrid and scalable portfolio of additional techniques. The result remains simple enough to be maintainable, comprehensible for the code writers as well as the end-users, and robust enough to handle a wide spectrum of possible mission scenarios and circumstances without modification.

1. Introduction

"There is no substitute for a human opponent." — Vincent "T.J." Taijeron, USMA Warfighting Simulation Center, West Point, NY. When one is lacking, however, we attempt to offer a usable substitute. In this paper, we describe an architecture and a methodology for software engineering a Computer Opponent Artificial Intelligence (COAI) for professional military training. Such an architecture ideally should meet the design goals of being frugal and efficient in code, easily maintainable, and produce an acceptable level of realism and flexibility for military training personnel and administrators. A truly low-overhead and low-impact solution to the vexing "AI" problem for professional military training at echelons below division and corps level. At the current time, a paucity of software exists, either commercial off-the-shelf computer games or DoD produced and acquired software, for low-cost training in this regard [1].

1.1. Background

Army simulation training has typically used extremely complex, sophisticated, and costly software that necessitates set-up time and planning, large staffs, large budgets, training, and Herculean scenario design efforts. Recently, there has been a shifting of emphasis to what is called low-overhead/low-impact computerized training that lower-level echelons, which traditionally did not have access to large-scale simulation support, can utilize effectively and efficiently. The offerings in this area are slim, and typically commercial computer wargames are wholly inadequate for many reasons. In particular, realistic and useful computer opponent "AI's" are virtually completely lacking. Tasking organization staff to "play the part" of opposing forces is a plausible solution, but necessarily involves a huge commitment of resources when theoretically the CPU can be doing the same thing at little to no cost. Architecting and implementing a targeted solution to the "AI" problem at the appropriate level of simulation and modeling fidelity has been a persistent issue for more than a decade [2,3].

1.2. Current Practice

Large-scale simulation exercises are frequently conducted at the higher levels of the Army command structure. These include division, corps, and army echelon levels. Lower-level training has largely been restricted to manual map exercises, or expensive field training and wargames [4]. Software tools at the company, battalion, and brigade training levels have been sparse. One frequently utilized piece of software is the VBS simulation, which

*Corresponding Author: Michael Pelosi, Professor, East Central University, 1100 E 14th St, Ada, OK 74820, USA | Email: michael@mauisolarsoftware.com

is commercially marketed as "Arma". This is a first-person shooter type game that has been utilized for squad and platoon level infantry type training. However, at the next higher echelons software training tools are virtually nonexistent. Even the widely utilized VBS platform has a woefully inadequate AI — routinely cooks and mechanics are tasked to drive trucks, fly planes and helicopters, and play civilians, as part of the simulation exercise.

1.3. Envisioning the Solution

How can the computer opponent AI problem be solved? Several AI approaches are possible. Hierarchical Task Network planning could be utilized. Unfortunately, HTN planning would typically result in one monolithic game-play solution largely predictable between sessions. Expert systems are possible, but structuring the implementation and gathering expert responses to all possible inputs is difficult and time consuming. Computer chess type solutions offer great promise, but it took over 50 years to get a computer to play a good enough chess to beat the best human players.

1.4. The Computer Chess Analogy

Despite 50 years of advancements in computer technology, computer chess AI still relies on a largely brute force approach. Using the Min-Max algorithm, transposition tables, and other optimizations, the chess AI scans through the game tree, analyzing millions of potential moves, before producing the highest scored next move [5,6]. Thankfully for chess, there are only 64 squares and a maximum of 32 pieces. In a professional military simulation, a map may consist of millions of individual 100-meter sized grid squares, thousands of units, and a completely unpredictable terrain and mission. In other words, searching through the game tree of all possible actions is tens of magnitudes more difficult. The chess modeling approach is not scalable, not nearly so. The chess analogy AI cannot be made to fit, yet we can adopt some of the lessons learned from computer chess. This includes dividing the session into phases: opening, middle-game, and endgame. Scoring the value of different pieces (maneuver unit groups), and evaluating final desired states, and how to get there, have proven to be extremely useful concepts.

1.5. A Scalable Hybrid Approach

The approach we have implemented and described in this paper could be considered as a hybrid AI approach. Relying on the chess analogy and metaphor, particularly the game phase concept, we add on an expert system that models and abstracts the actual units in decision-making processes wherever possible. For example, the AI groups subunits into their actual mission structure, including companies and battalions. These are controlled as in the military force structure chain of command. For the creation of plans, it is possible to adopt the actual military staff procedure and adapt this into a software planning sequence. Further, lower-level planning details such as route determination can be supplemented by the ubiquitous A* Algorithm. Through trial and error over a development lifecycle of several years, and through feedback from professional users, we have cobbled together a hybrid approach that is efficient, robust, and easily implemented. Flexibility for future features and maintainability has been realized.

1.6. Presentation Goals

Our goals in this presentation are to describe to the simulation community, and perhaps computer gaming community, the derived computer opponent AI architecture. More than 10 years of effort have been devoted to its creation. Much of this time was spent ruminating on the final ideal implementation — a big part of the solution was multiple layers of modeling abstraction. Isolation between the layers was and is also extremely important for comprehension and maintainability. Further, and to a large degree, the AI layers actually model in parallel the military command echelons.

1.7. Engineering the New Approach

There are individuals who spend vast portions of their waking hours operating and contending with AI code in simulations and games. We have been careful to consider their recommendations and concerns in engineering an appropriate and efficient AI system for land warfare simulation. One of the quotes that comes to mind is the following: "The AI must perform the mission, the AI must be a good soldier not a general." In other words, unexpected or completely unpredictable behavior is out of the question. This is both for realism and utility. If the trainee utilizing the AI cannot understand what it is doing it may be time wasted. In addition, the scenario designer may want to script certain actions to instruct certain doctrinal principles. Thus, scripting of the AI should also make it possible to funnel behavior into a certain group of possible actions. At the same time, the AI cannot be overly complex: "…and one turn requires 15 minutes of preparation and then crashes! Good job!" Clearly there is a trade-off between complexity and robustness.

2. Methodology

A realistic AI useful for professional training purposes should both model and mimic the military decision-making process at various echelons. In that light, as a robust foundation the AI goes straight to the U.S. Army field manuals for guidance. Fortuitously, more than 100 years of modern warfighting experience has distilled down Army planning doctrine to a few formulaic processes in the Military Decision-Making Procedures (MDMP). These include TLP, METT-TC, and OCOKA. These processes can and have been modeled almost directly in code.

2.1. Military Decision-Making Procedures

U.S. Army Field Manual 101–5, Staff Organization and Operations, explains in detail the Army MDMP (U.S. Army 1997). "The MDMP is an adaptation of the Army's analytical approach to problem solving. The MDMP is a tool that assists commanders and staff in developing estimates and plans. The full MDMP is a detailed, deliberate, sequential, and staff-intensive process used when adequate planning time and sufficient staff support are available to thoroughly examine numerous friendly and enemy courses of action (COAs). This staff effort has one objective—to collectively integrate information with sound doctrine and technical competence to assist the commander (in our case the COAI "commander") in decisions, leading ultimately to effective plans. The analytical aspects of the MDMP continues at all levels during operations."
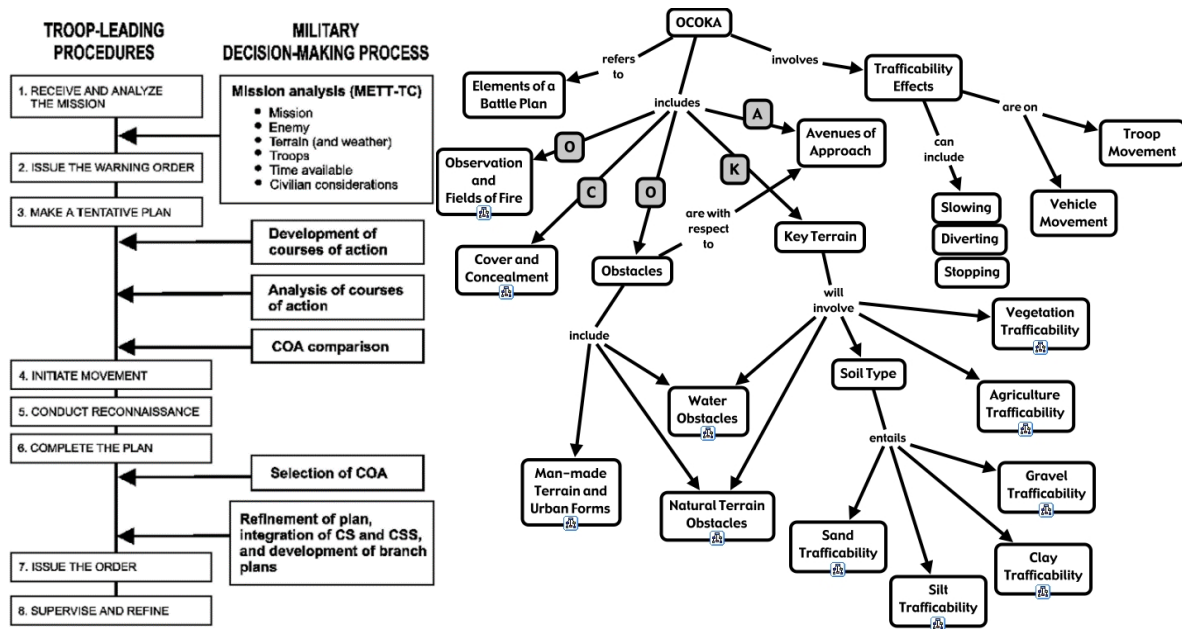
Figure 1: TLP, MDMP, METT-TC, and OCOKA Depictions [7]

2.1.1. Military Missions and Objectives

The COAI is presented a military mission that is contained within scenario and AI option specification files. The files include information on task organization, friendly forces, and a timeline. Objectives are also specified with points values for various objective type such as occupying a location, clearing an area of enemy forces, moving friendly forces past a certain demarcation zone, or searching for a hidden target.

2.1.2. Military Decision Making Procedures

Where possible the military decision-making process (MDMP) is then followed both in modeling and implementation. The COAI is designed to follow a similar process to what is recommended doctrine in the Army field manuals. Likewise, parallel modeling and decision-making takes place at each of the important unit echelon levels: platoon, company, battalion or task-force, and support. The following further outlines MDMP aspects:

TLP (*Troop Leading Procedures*) consist of the following steps: 1. receive the mission and conduct METT-TC and OCOKA, 2. prepare for the mission and issue preliminary orders, 3. make a tentative plan: identify goals, gather information, generate/analyze/compare possible solutions, and implement the best tentative plan, 4. start movement, 5. conduct reconnaissance, and 6. follow through with execution of the final plan.

METT-TC is: *M*ission analysis, *E*nemy analysis, *T*errain analysis, *T*roops analysis, *T*ime limit analysis, *C*ivilian impact analysis. *OCOKA* is conducted as part of terrain analysis.

OCOKA stands for *O*bservation and fields of fire, *C*over and concealment, *O*bstacles, *K*ey terrain, and *A*venues of approach. This constitutes a more detailed terrain analysis. Obstacles can include man-made and urban terrain obstacles, natural terrain obstacles, and water obstacles. Key terrain may involve, for example, high elevation or easily traversed terrain near objectives. Avenues of approach include roads and otherwise clear areas.

Trafficability can be evaluated both for vehicle and troop movement regarding slowing, diverting, or stopping movement.

2.1.3. Course of Action Adoption

The Course of Action (COA) for the COAI that is produced is a result of the analysis of the above factors and constraints. Reducing all considerations to a quantitative scoring allows a brute force solution, that randomly generates various plans and each plan can be scored for its suitability and feasibility. The highest scoring feasible plan is selected as the best COA. Since plans are randomly generated, differing and unique plans are generated during each new instance of the same scenario mission design. This is important for replay value.

Execution requires close supervision and monitoring, as well as continuous analysis, the updating of intelligence, and refinement of the COA plan. In certain cases, the plan must be discarded and regenerated completely.

The COAI has certain unit groups assigned to it in the scenario and mission design. This allows the possibility of multiple instances of the AI, each controlling its own respective force grouping. Likewise, human participants would each be controlling various force groupings.

2.2. Mission Analysis

Scenario design inputs to the COAI for information analysis include: time limit constraints, enemy OOB, friendly OOB, quantified scenario objectives, along with AI options and settings. Five major phases are accomplished during a preliminary mission analysis:

1. Analysis and calculation of the goal state to satisfy objectives. Often this will involve the ideal placement of forces by the scenario end time, such as the occupation of objective locations.
2. Analysis and calculation of known enemy dispositions and force allocations. Here relative points values are calculated

relying on "combat power" summations for known enemy unit types in a catalog database of unit types. Friendly unit group combat power totals are likewise analyzed to create favorable force match ups. In general, a ≥ 3:1 points total advantage will be necessary for successfully taking the occupation of locational objective from a defender [8].

3. Analysis and calculation of a tentative plan to reach the goal end state. This is accomplished by using a brute force approach like the solving of the Traveling Sales Person problem [9]. Several thousand likely plans are randomly generated and scored. Top scoring plans are then further evaluated and selected.

4. Surplus time and resources are evaluated. If the plan can be accomplished before the scenario mission end time, further refinements and optimizations can be preliminarily executed. This can include actions such as further intelligence gathering and reconnaissance, softening up of target locations through preliminary bombardments and airstrikes, and conducting feign attacks or deep pincer movements.

5. Finally, initial tentative movements and actions for the first game phase are calculated, however these may change when the final COA is adopted.
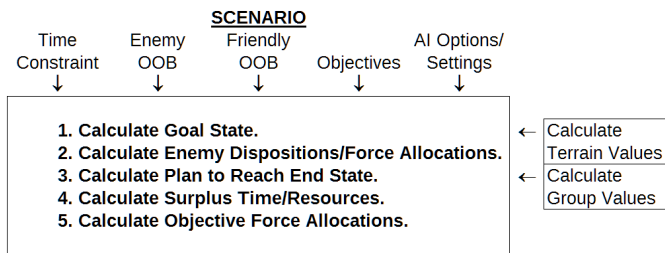

Figure 2: Scenario and Mission Specifications and Analysis.

2.2.1. Discrete Mapping Zones

As part of the mission analysis and execution, map zones are segmented and demarcated. Map zones are segmented based on the center mass coordinates of friendly forces, and known or likely center mass coordinates of enemy forces. With these two points fixed in space, a relative "mapping" of center, left and right flanks, and depth can take place. If portions of these areas are off the given terrain area, they are ignored and become notional. Areas which are untrafficable (for vehicular and/or foot units respectively) are also ignored for deployment or movements. The point halfway between friendly center mass and enemy center mass becomes the Forward Edge of the Battle Area (FEBA) anchor. Ten discrete zones are then demarcated from this mapping: forward screen left, left, center, right, screen right, and corresponding rear areas respectively. Reconnaissance missions will be routed into the far side of the FEBA, and missions will be allocated to specific zones. Combat group deployments take place generally in the left, center, and right, with screening units assigned to screen left and screen right areas. Supporting units, including headquarters, artillery, and logistics, are routed toward the rear areas. Spacing between units is calculated by the frontage span of the respective maps zone. Reserves are held in the rear area as well. Thus, any map size from 5 km × 5 km up to 500 km × 500 km can be automatically mapped into a convenient "scenario" mission and planning space, based on the map size, force size, and initial deployments.

2.2.2. Map and Terrain Analysis and Updating

As part of the mission analysis, a detailed terrain analysis is conducted and the results are stored in a map database of grid squares. Each map grid square is assigned a weighted and then normalized score value for specific characteristics. These include the following:

Map static terrain analyses:

- *Objectives* – value and proximity to objective locations.
- *Water* – water obstructions to ground movement — note this may have an important lack of effect on the many amphibious vehicles in operation.
- *Elevations* – in many circumstances higher elevation locations are more valuable to occupy.
- *Grades* – steep uphill and downhill grade serve as detriments to mobility.
- *LOS* – line-of-sight to nearby grid squares; some locations can observe much more of the surrounding terrain.
- *Blocks* – blocks can include highly dense vegetation, urban locations, as well as man-made obstacles.
- *Cover* – cover provides shelter from blast effects and observation.
- *Avenues* – key avenues for movement, central locations networks to objectives are preferred.
- *Concealment* – concealment has low line of sight visibility as well as good cover.
- *Defense* – combination of effects from above for defensibility.
- *Survivability* – cover, concealment, and defensibility modified for survivability aspects.
- *Ambush* – areas with good visibility/survivability, nearby to key avenues of movement.
- *Counter mobility* – places where the enemy's movement can be stopped efficiently.

Map dynamically updated analyses:

- *Valuable Areas* – avenues, high elevations, nearby to objectives, etc.
- *Friendly Proximity* – weighted/normalized value for staying close to friendly concentrations.
- *Enemy Proximity* – weighted/normalized value for known/updated enemy concentrations.

The COAI user interface produces shaded map graphics depicting weighted and normalized values for each of the terrain analyses based on grid square location. The last three terrain analyses are dynamically updated as the simulation progresses, and reflect new and current information.

2.2.3. Order of Battle (OOB) Analysis

Because of the COAI not considering individual unit entities at the lowest level (platoon and section sized entities), the COAI is only aware of the unit groupings, relative combat power, and the group types. Using these characteristics, Order of Battle (OOB) analysis can assign various groups to specific objective missions. For example, an armor company may only constitute 14 tanks but have triple the combat power of a 90-soldier infantry company. The combat power is based on points totals from the unit catalogs. As part of the COA production, the COAI analyzes the most

optimal assignment of groups to objectives using its limited knowledge. With regards to enemy forces — given adequate knowledge, OOB analysis will endeavor to produce adequate match ups, specifically the greater than 3 to 1 advantage of an attacker over a defender in terms of combat power.

2.2.4. Spot Table and Updates

The COAI must keep track of known and likely enemy force locations. Initially, the training scenario designer can choose to reveal as much or as little about enemy force locations as desired. This can be loaded into an initial enemy spot table at scenario start, and be used for COAI planning. After that the COAI is on its own gleaning, updating, and aging information as it comes in. It keeps track of this in a dynamic spot table that contains coordinates, unit type and points value, and most recent spot time. As spots are aged they are reduced in weighting importance for relevance and accuracy. The simulation produces a listing of current force enemy spots and hands it off to the COAI, which collates and posts the information in the spot table.

2.2.5. Force Allocations

As previously mentioned, planning the force allocations of groups to objectives involves a brute force planning approach. For each of several thousand plan iterations, groups are randomly assigned to objectives. Then, based on the group data, objective data, and enemy locations, a special function calculates the feasibility of each allocation. Time to reach the objective, attack or defense ratio, and other factors, can cull out infeasible assignments. Of the remaining feasible plans, these are scored based on minimum cost in terms of movement time, and attractiveness of force match ups (desired minimum 3:1 on offense) and other factors. A final "minimum time to complete the plan" duration is calculated, and from this any surplus time available for further measures, such as reconnaissance or softening up of targets, is then known. Force allocation intrinsically calculates the end game phase plan. Since each scenario objective is assigned a relative points value, scoring of plans takes into consideration the satisfaction of more valuable objectives, as well as the distance from each respective group to its assigned objective. For further information on brute force planning using this approach, traveling salesperson solutions are a good starting point [9].

2.2.6. Course of Action Decisions, Types, and "Playbooks"

Once final endgame force allocations have been favorably calculated, if sufficient surplus time and resources exist, a middle game "playbook" COA can be adopted by the COAI to further shift the favorable odds preliminary to the endgame phase. In the case of a defensive posture this can include securing objectives, static defense, or in-depth defense [10]. For attack postures, broad front attacks, counterattacks, or deep attack "playbooks" can be adopted. A reserve force can possibly be selected. The playbook selected is not optimal, but suitable for the given situation and circumstances. This is analogous to the football play: a running play may not be any better than a deep pass, but it keeps the other team guessing. Seemingly random intelligent plans and actions in terms of time and execution are an important part of a realistic and engaging COAI with suitable replay value.

2.3. Major "Game" Execution Phases

Once the COA has been finalized execution will be transitioned through a series of major game phases, relying on the chess motif. These include the opening, middle-game, and endgame. Assuming little slack time exists for achieving the mission objectives — the execution phase will be shifted immediately to "endgame". Endgame can be considered the all-out effort to achieve the objectives immediately. Otherwise, if slack time exists, perhaps an opening and middle-game phase will be adopted as part of the execution.

2.3.1. The OODA Loop: Observe, Orient, Decide, and Act

The execution phases are analogous to the very important military precept of the OODA loop [11]. The opening is comparable to the *O*bserving phase. Here, advantages are to be acquired in terms of additional intelligence and other measures, such as occupation of key terrain. Middle game is analogous to *O*rienting — the major reorienting of friendly forces to further tip the balance for further movements and attacks. Endgame is the *D*ecide and *A*ct portion, where commitment to a decisive outcome is adopted. Final actions are wagered here. Replanning is necessitated between the opening phase and the middle-game phase, as well as between the middle-game and endgame phase.

2.3.2. Execution Phase Transitions

Transitions between the major game phases are based on the characteristics of what should be taking place generally in each phase — once again relying on the chess metaphor. The opening takes place between the scenario start and first enemy contact, first weapons fire, and/or first friendly casualties. The endgame is transitioned to after the middle-game when the previously calculated time deadline for accomplishing the mission objectives is reached. This also includes a time safety factor built-in. In other words, final execution is committed to when there is still enough time to safely accomplish the objectives. That said, to preserve verisimilitude, there exists the possibility of a "lightning battle" COA adoption where the COAI will skip the opening and/or middle-game phases and progress directly to an endgame phase. This is analogous to a surprise execution, which forces the training audience to consider all possibilities. Skipping phases is easily incorporated into the COAI options as probability factors for skipping middle game, and skipping opening and middle-game. At the juncture of each major game phase, replanning takes place based on the phase goals described further below.

2.3.3. The Opening Phase

The opening phase is largely characterized by observing the enemy's respective force deployments and dispositions. Goals here include grouping and further deploying friendly forces, exploiting terrain based on mission analysis terrain calculations, reconnaissance missions are executed, counter reconnaissance missions are executed, and the seizing of easy objectives closer than the enemy which may not be occupied.

2.3.4. The Middle-Game Phase

The middle-game is characterized by major force movements to orient deployment for final attacks and/or defense. Seizing of key intermediate terrain is conducted. Harassment missions are

Transitions	Execution Phases	Mission Replanning	OODA:	Phase
		Scenario/Terrain/Objective Analysis		
	OPENING			
Scenario Start	·Group/Deploy Forces ·Exploit Terrain ·Recon ·Counter-Recon ·Seize Objectives Closer than Enemy	**Opening COA Generation** *Precept: Groups become independent agents after being assigned a mission and can replan as desired.* *Independent Mission Replans/Spawning*	*OBSERVE*	Prob Skip ~ 15%
	MIDDLE-GAME			
First Contact *First Fire* *First Casualty*	·Major Movements ·Seize Key Intermediate Terrain ·Allocate Resources to Endgame ·Hold Reserve/Counterattack Force ·Counterattack ·Find-Fix-Flank-Finish	**Middle-Game COA Generation** *Independent Mission Replans/Spawning*	*ORIENT*	Prob Skip ~ 25% / Surplus Time/Resources
	END-GAME			
Transition to Final Maneuver/ Attacks *(Required Time + Safety Factor)*	·Regroup ·Final Attacked/Objectives ·Execute	**End-Game COA Generation** *Independent Mission Replans/Spawning*	*DECIDE ACT*	Minimum Time to Accomplish + Safety Factor

Figure 3. COAI Execution Phases, and major aspects of each phase.

perhaps selected, these would include randomized probes or artillery fire missions. Allocation of resources to the endgame is recalculated. Further, most COAs will hold a major reserve and/or counterattack force for unforeseen events. Counterattacks can also take place. Attacks use as a basis the "4F's" for planning and execution: *find – fix – flank – finish* [6].

2.3.5. The End Game Phase

The endgame embarks on achieving the final scenario objectives. Thus, a regrouping of scattered forces may be necessary for the execution of the final plan. Final attacks are enacted, if necessary, and objectives are occupied. The plan is irretrievably executed at this point — for either final success or failure. Ideally, the opening and middle-game phases have set up the COAI for uncontested victory at this point through incremental and methodical gaining of advantage. As mentioned, the endgame is analogous to the Decide and Act portions of the OODA loop, and opening and middle-game phases only take place if surplus time and resources exist for the satisfaction of the mission goals. Otherwise, the COAI would need to embark on an endgame plan immediately at the scenario start.

2.4. Modeling Layer Architecture

Analogous to the OSI model [12] for computer networking (and its inherent division of responsibilities and functionality), there are at least six layers and levels of modeling that are used in the COAI architecture. Most of these parallel a corresponding layer in the military decision-making process and unit echelon structure, in real world military forces. This leverages the concept of object modeling for real-world abstractions, as well as organizes and simplifies the architecture software code. At the lowest level are the simulation entities, nominally platoon down to section sized organic units. The simulation in usage models each of these uniquely as a C++ class entity. Typically, these are grouped into company to battalion sized units, each consisting of 4 to 10 subunits. It is these groups that constitute the "unit groups" that are under direct control of a scripting engine layer. The scripting engine layer is responsible for issuing the entire group

order directives discussed in more detail below. Above the scripting layer is mission control, more directly controlled by the COAI. Once the group has been assigned to a mission, the mission instance is responsible for autonomous control over the group through the scripting engine. Mission sequences, are in turn controlled by the overall COA class plan that is being implemented. And finally, the *COA class* is planned in response to the overall scenario objectives, available resources, game phase, terrain map, and options.

Table 1: Modeling Layers of Abstraction, Planning, and Control.

1.	Execution Phase: Open, Middle, Endgame → Determines strategic approach/goals.
2.	Course of Action (COA) → Self-contained master plan for phase, controls Layer 3.
3.	AI Mission Sequence Collection → Insertion, deletion, reordering possible.
4.	AI Independent Mission Control Agent → An autonomous OOP class, with reports.
5.	Group Scripting Engine → Programmed Sequences of Events/Actions/Responses.
6.	Unit Entity Grouping → Company/Battalion/Task Force, abstracts Layer 7.
7.	Simulation Entity → Platoon/Section/Section/Battery/Vehicle. Hi-fidelity modeling.

2.4.1. Section-to-Platoon-Sized Entities

Section to platoon entities are the lowest level of fidelity in the simulation in usage. The COAI does not control this echelon directly. The group and scripting engine issues direct orders and

Figure 4: Group Orders Scripting User Interface, which allows the inputting of sequences of orders.

commands to entities at this level. In summary, sections and platoons are characterized by locations, ammunition and fuel levels, strengths and casualty levels, current orders and status, among other data in a cornucopia of minutiae. Accurately modeling this spectrum of characteristics is an extremely labor-intensive task, that requires copious research and data entry. Accounting for hundreds of data items into COAI considerations is architecturally untenable, hence the abstraction to larger unit groupings is necessary to accomplish the goal of a robust and usable AI with a frugal amount of code. Codewise, these entities are modeled as classes.

2.4.2.   Unit Groups

Scenario design creates company and task force level unit groupings that normally model individual combat companies or battalions, artillery batteries, helicopter flights, and other unit groupings that would normally be controlled by a battalion or brigade level task force organization. Unit groups are modeled as a class and are the owner of combinations of the platoon and lower level entity grouping.

2.4.3.   Group Orders, SOPs, and Formations

Group orders and tasks are relatively straightforward and implemented easily by the controlling mission class. The controlling mission class merely instructs the scripting engine to calculate and implement the command order. Group orders contain such simplistic directives as: move *n* meters, change facing, set speed, dismount infantry, improve position, camouflage, discharge smoke, or set formation. Set formation automatically orients the group in, among others, line, column, box, diamond, forward wage, reverse wedge, and echelon formations. Company and battalion sized formations will typically orient themselves in one of the aforementioned formations to advantageously engage likely targets. The scripting engine handles the details, while the COAI concentrates on decision making at the next echelon above. Lower level units are responsible for handling their own engagement of targets of opportunity. Standard operating procedure (SOP) allows

independent decision-making for units in regards to firing smoke or vehicle engine exhaust smoke systems for defense, reversing on enemy sightings, or aggressively attacking new contacts.

2.4.4.   Group Scripting

Group order scripting consists of a collection of sequential orders. As mentioned earlier, movements, formation changes, camouflage orders, orders to "dig-in", etc., can be added to a scripting sequence. The scripting engine automatically executes the orders serially until completion. The COAI process communicates into the simulation the desired scripting sequence and parameters through active mission classes. The user interface of the group orders scripting window is shown below in Figure 4. Scripting can be manually controlled as desired, and saved to file.

2.4.5.   COAI Control

The COAI enters a control main loop after completing the mission analysis and creating a preliminary COA. Each time the loop executes, more information is extracted from the simulation, this is processed, and the COAI may modify directives or give additional orders to the group scripting engine. Groups with scripting orders pending can have those sequences cleared if necessary. The main loop continues until the scenario end time is reached and objective condition scores calculated.

2.4.6.   Course of Action (COA) Modeling

COAs are modeled as a class and store their own data and update themselves inherently. Additionally, given certain circumstances they are capable of canceling themselves which will necessitate and involve an automatic regeneration of a COA. This can optionally be done at random, or in the case of catastrophic goal failure. COAs contain enough information to be considered a high-level plan, with very little implementation details.

2.5. Mission Types

In satisfaction of the current COA, unit groups are assigned sequences of missions that fall into various categories. Each of the

missions is defined in a C++ class, and the sequences of missions are collections of missions. The unit group conducts the next mission listed in its respectively assigned mission collection, until each one is completed. If necessary, a new mission can be spawned and inserted at the top of the collection, at which time the unit grouping will embark on the new mission, and resume the second mission once the newly spawned mission has been completed. For example, a grouping on a movement mission toward an objective can be assigned a newly spawned mission to attack a target of opportunity. Once this attack mission has been completed, the movement toward the objective mission will be resumed. Further, since missions are modeled as autonomous agents, they can spawn their own new missions as necessary which may supersede the current mission. The hierarchical breakdown of various COAI mission classes developed totals over 40 at the present time.

2.5.1. Mission Agents (as Self-Planning, Autonomous, Self-Updating, and Reporting OOP Classes)

As mentioned, missions are modeled as classes and have a decoupled implementation. Each mission has a pair of classes closely related: a planner class and an implementer class. The planner class plans the mission and hands over the implementation details to the implementer class. If the implementer class runs into problems, it will call the planner class to once again reinitialize and replan the mission. Once a mission is spawned, it is initialized with several goal variables and the mission class code itself calculates how to correctly carry out the mission. During each iteration update (periodically calculated based on an AI update time step), the mission updates itself and commands to the mission unit grouping as necessary. Upon mission completion, the mission class instance is removed from the mission sequence collection and ceases to exist. Some of the major mission taxonomy types, which rely on C++ class inheritance from the mission base class, include movement missions (which activate A* pathfinding), attack, defense, recon, and support missions. Missions are queryable for public properties such as mission start time, estimated completion time, status codes, and other information. As a result, it is straightforward to keep the user interface updated with graphical status for users.

2.5.2. Movement

Movement missions are generally tactical movement or road movement missions. Tactical movement will move in a tactical formation using advantageous routes to the goal endpoint. Road movement will simply travel by the most trafficable route to the goal location. Generally, movement missions will respond to RTC (React To Contact) events using a SOP, which may include attack, evade, stop movement, or retreat doctrines. Recon missions are similar but will move to advantageous locations for observation based on the terrain analysis, among other doctrinal differences.

2.5.3. Pathfinding

Pathfinding to mission waypoints along a movement route is calculated using a modified version of the A* [13]. The implementation considers multiple goals, for example the cost function can include factors for avoiding or approaching the enemy, attractiveness for traveling on roads, moving through high line-of-sight grid squares, or maintaining terrain cover. For example, if one of the mission goals is concealment during movement, lower movement cost can be assigned for terrain covered by forested areas or buildings. Pathfinder estimated time of arrival results are based on the speed over terrain of the slowest unit in the group.

An important consideration to note is that pathfinding algorithm code must take place in its own CPU thread. As a result, when a mission needs a pathfinding route, it sends desired goal coordinates as well as group data and route preference to a collection of pathfinder processes executing on the machine. The pathfinding request is queued and the mission class waits until a result is returned. Pathfinding is by far the most computationally intensive element of the entire COAI architecture, other than initial scenario and terrain analysis. Route movement of mission groups is 80% of what the COAI does. The importance of this cannot be under-stressed: a brick-house architecture for planning and implementing movement has been essential.

2.5.4. Attack Missions

Various type of attack missions are implemented based on group composition; vehicular, foot, aircraft, etc. Generally for attack missions an endpoint location is assigned as well as a casualty threshold. The group will generally conduct tactical movement toward the objective, execute the attack several times as necessary, and regroup between attacks. During movement react-to-contact standard operating procedure is enacted.

2.5.5. Attack Precept: Find, Fix, Flank, and Finish

Generally, attacks are coded to take place using the well-established military metaphor for success which elaborates on the "4F's": find 'em, fix 'em, flank 'em, and finish 'em [14]. Therefore, planned attacks would normally involve coordinated movements and attacks by two or more unit groupings, comprising at least a $\geq$ 3:1 combat power points advantage. Surprisingly, attacks are tractable to plan with acceptable realism once an enemy defender has been located. The fixing force approaches directly within weapons distance and begins firing. Meanwhile, the flanking force(s) conduct flanking movements around the left, right, or rear, and engage the enemy from that direction. The finishing force can be either the flanking force or another available group, which will move in for a final clobbering. A casualty loss threshold is established preliminarily, and if it is reached the attack will be broken off as unsuccessful and the forces reallocated.

2.5.6. Defend Missions

Isolated group defend missions are more simple and will merely move to the endpoint location and prepare a defense, normally by "digging in". If a casualty threshold is met the group will withdraw to a safer location.

2.5.7. Support Missions

Support missions include being held in reserve, artillery fire support, logistics and supply, and headquarters missions. Forces allocated to support missions, in addition to conducting their primary mission, will relocate periodically to maintain a relative position behind the FEBA. Most support missions will also
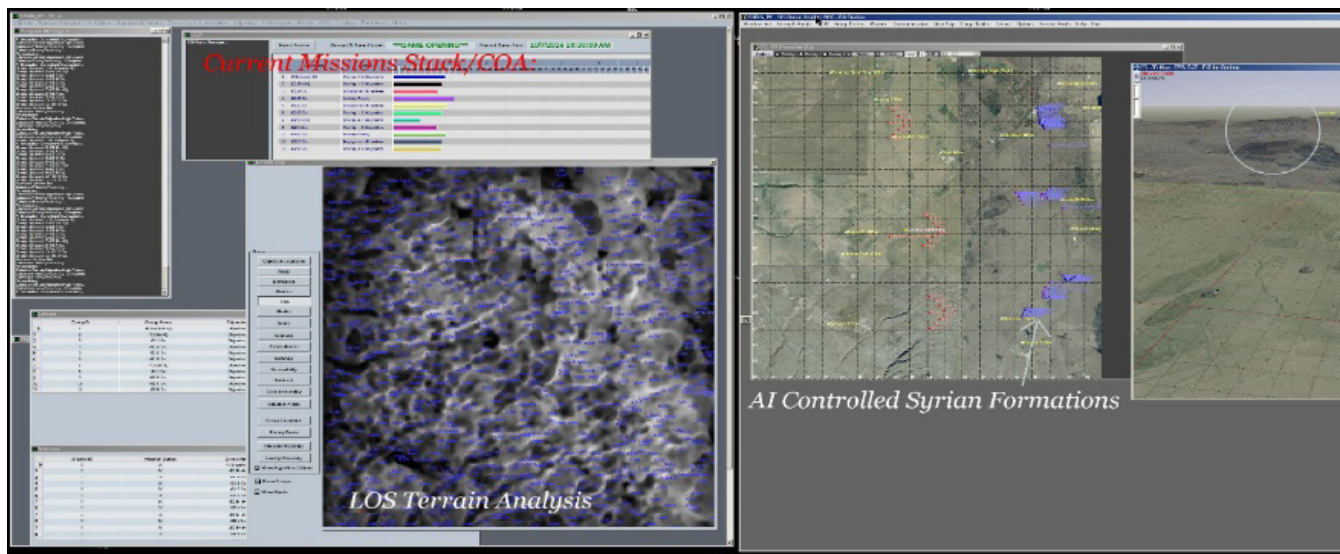
Figure 5: COAI User Interface with mission Gantt chart, terrain maps, and mission status logs.

periodically relocate based on their proximity to the enemy, for example headquarters units will maintain a safe distance between themselves and the nearest enemy, or advance to maintain a general distance to the FEBA. Artillery monitors the current spot table and fires on lucrative targets of opportunity, or fires in support of attacking or defending units when advantageous.

In the special cases of artillery and attack aviation support missions, available direct fire artillery as well as available attack aviation assets, are placed in pools. Group missions can request fire or aviation support based on their circumstances, in which case it is added to a request listing. Artillery and aviation assets periodically evaluate the listing and prioritize their response based on likely effectiveness and proximity. Once satisfied or determined infeasible, requests are removed from the queue listing.

2.6. Group Force Types

Missions are assigned based on unit group type, which is known from the scenario design specification. In general, most unit groupings can be categorized into one of the following major groups: armor, mechanized, mechanized with dismounts, infantry, artillery, recon, screening, aviation (attack/transport/recon), refueling support, ammunition support, or headquarters. The sum of the combat power points for various units is used for calculation of overall combat power and force match up ratios. Periodically, groups may find themselves unassigned to any particular mission. In this case the COA class will score and produce their next best mission assignment.

2.7. Software Implementation

The adopted architecture by necessity uses an object-oriented language. Prolific usage of C++ classes and inheritance features are realized. In particular, the ability to create object collections is critical, as well as the ability for data hiding, decoupling, and multi-threading.

2.7.1. Code Architecture

It is a very important design principle that the COAI code takes place in a separate process in isolation from simulation code.

This allows a very important decoupling and higher-level abstraction from the lower-level details generally inherent in the simulation. Additionally, the general time step in the simulation is a tiny fraction of the time step necessary for the COAI calculation and update. For example, if the simulation time step is 30 seconds per iteration, AI is easily updated at 10 minute intervals.

As mentioned, COAs are implemented as a class object, and own mission sequence collections of mission classes. Each unit group has its own mission sequence collection. The current mission is the mission on the first position in the mission sequence collection. This mission class is responsible for controlling its assigned mission group. It passes high-level scripting orders to the simulation scripting interface. The scripting interface controls sequences of orders which include items such as move to waypoint, change formation, change facing, move at speed, and weapons tight or weapons free, among others.

2.7.2. Sequence Execution

When the COAI process is initialized with data, it begins scenario analysis, terrain analysis, and OOB analysis. Typically, this process can take several minutes. When preliminary analysis is completed, an endgame COA is produced as well as the group maneuver plan. Assuming surplus time and resources exist, an opening and/or middle-game COA may be implemented instead. At this point the COA is formally implemented, missions are spawned (which are responsible for planning their own independent execution, and updating themselves, or spawning new missions as replacements). Missions are then implemented. Current missions are posted to a Gantt chart in the user interface which includes each unit grouping. The Gantt chart depicts the mission stack for each group, as well as an estimated completion time for respective missions. The COA is then controlled until the next game phase is reached. The scenario ends at the scenario end time and the victor is calculated based on objective points totals.

2.7.3. COAI User Interface and Processes

It is important to give the training audience and training administrators a detailed window into the internal workings of the

COAI; this is both so that they can understand it as well as appreciate the realism inherent in the modeling. Further, they can more intelligently tweak the COAI settings and options and locate defects and probable improvements. The COAI has its own process independent UI running in parallel with simulation code. Some of the available graphical interfaces include an event log, static terrain map, updated maps with group locations, objective table listing, COAI options, and scenario options windows.

User interface values are updated at each calculation iteration of the COAI, typically at one minute or greater intervals (clock time). The current game time and phase, and information on the COA in execution, is also displayed. It is possible to bring up child windows displaying detailed information on missions, along with corresponding mission log windows. For debugging, it has been necessary to display pathfinding windows with the status of individual route planning efforts.

The user has the option to manually toggle the next game phase, and go from opening to middle-game, or middle-game to endgame. Further, the user can manually toggle a complete COA replan for the current game phase. Figure 5 below shows the COAI user interface.

2.7.4. Allowable Matchups: H2H, H2AI, AI2AI, and MH2MAI

Flexibility for opponents is maintained. Human versus AI contests are possible, as well as AI versus AI, multiple human teams versus AI, and multiple human teams versus multiple AI instances. All that is necessary is instantiation of a unique simulation user interface, and the AI interface that sits on top of it. AI implementation is basically built on top of a simulation user interface. The COAI takes advantage of and issues UI controls and commands not unlike a human user.

2.7.5. Incorporating Plan and Action Randomness

Is important to note that the AI solution to the scenario problem does not have to be optimal. In order to be realistic and acceptable as a computer opponent, it is well known that human opponents are far from optimal. However, they can be counted on for a unique solution to most specific problems that will vary significantly between occurrences. If optimal mission solutions were calculated, this would result in a decreased training benefit and replay value for the AI implementation. Using randomness, crucially, also greatly simplifies the coding and modeling requirement necessary. Missions can be randomly specified within certain acceptable limits; in type, time, and space. For example, a recon mission can set a goal 0.9 km distant from a similar mission, and the overall results will not be that much different or unrealistic. A battalion can attack at 21:00 hours instead of 19:30 hours, and the effect is perhaps more realistic than if all attacks took place at 19:30 hours in similar circumstances.

2.8. Example Execution

As an example of a typical execution, assume a mission with three objectives: A, B, and C. The AI has 5 unit groupings available, and conducts a preliminary mission and terrain analysis resulting in an "endgame" COA. In this COA the 5 unit groupings would proceed directly to their respective objective, and would be assigned using the brute force plan generation methodology

previously described. Since several hours of surplus time is found to exist before the scenario deadline, the AI embarks on an opening and middle game phase. During this phase reconnaissance is conducted as well as the seizing of key terrain, as calculated by the terrain analysis. Upon first enemy contact or casualties, the AI transitions into a middle-game phase and conducts preliminary attacks and deeper movements, preparing for final endgame execution. Once the time threshold for proceeding into the endgame phase is reached, the AI transitions into the final execution and replans the endgame movements from the current situation. Forces are allocated and final movements and attacks are conducted.

3.   Results and Conclusions

The model described in this paper constitutes a computer opponent AI system conceived for the conducting of professional military training. Although many approaches exist, it is important that the envisioned solution be capable of a low-overhead implementation in usage. In other words, a large stimulation staff and large budget must not be required for end-users to conduct their own training. No special hardware, facilities, or preparation must be required. Bringing low-cost computer-assisted training to the echelon targeted (company, battalion, and brigade level training) requires this characteristic.

3.1. Advantages

The approach outlined has been able to achieve a robust and realistic computer opponent, while at the same time maintaining a reasonable level of coding overhead and debugging requirement. Much of this is due to prolific abstraction and layer isolation. The implementation as a result remains simple enough to understand at each layer, and exists in segregation from the decoupled simulation engine. Grafting a user interface over the AI has allowed users to understand what is going on and to better tweak settings and provide improvement recommendations and defect reports. The AI produces significant enough mission solution differences between instances to allow valuable replay value of identical scenarios.

3.2. Disadvantages

Inability to do a major replans during one of the transitional game phases intelligently is perhaps a disadvantage. Once the AI commits to a certain plan in any of the major game phases, only a triggered event can cause a major replan. In other words, tweaking the plan significantly does not happen within a respective game phase. Certain thresholds can be set, such as an overall casualty factor (as in the event of a catastrophic plan failure) in order to reinstitute a new "ground-up" planned COA. However this is not completely congruent with human level decision-making. Ironically, it may come close — and close enough is what we have been after. Another disadvantage is inflexible unit groupings. Generally, the past grouping that the scenario designer has created in the design files is what the AI will have to implement in force allocations.

3.3. Future Work

Future work includes the addition of more mission types and implementing them in respective classes. Further refinement of

the COA production process and the addition of wider spectrum of playbook templates is envisioned. Better coordination between attacking unit groups is also necessary. This may require an additional layer of abstraction constituting a class (below the COA class level) that controls multiple mission classes — this layer of modeling does not yet exist. Incorporating genetic algorithms and neural networks into mission planning is under investigation. Additionally, implementing graphical output depicting mission maps and plans according to guidelines put forth in specifications outlined in FM101-5-1 Staff Organization and Operations [15] is desired. Black Box AI is a thing of the past. The audience deserves and should demand Clear Boxes.

References

[1] M. J. Pelosi and M. S. Brown, "Software engineering a multi-layer and scalable autonomous forces "A.I." for professional military training," 2016 Winter Simulation Conference (WSC), Washington, DC, 2016, pp. 3122-3133. doi: 10.1109/WSC.2016.7822345

[2] Johnston, J. H., Goodwin, G., Moss, J., Sottilare, R., Ososky, S., Cruz, D., & Graesser, A. (2015). Effectiveness Evaluation Tools and Methods for Adaptive Training and Education in Support of the US Army Learning Model: Research Outline (No. ARL-SR-0333). Army Research Lab, Aberdeen Proving Grouond, MD.

[3] Lane, H., Core, M., van Lent, M., Solomon, S., and Gomboc, D. (2005) *"Explainable Artificial Intelligence for Training and Tutoring."* 12th International Conference on Artificial Intelligence in Education, Amsterdam, The Netherlands.

[4] U.S. Army (2003). *"FM 7–1, Battle Focused Training*." Washington, DC: GPO.

[5] Newborn, M. (2012). Kasparov versus Deep Blue: Computer chess comes of age. Springer Science & Business Media.

[6] Shannon, Claude (1950). *"Programming a Computer for Playing Chess"*, Philosophical Magazine 41 (314)

[7] U.S. Army (1997). "*FM 101–5, Staff Organization and Operations".* Washington, DC: GPO.

[8] U.S. Army (2002). "*FM 3–06.11, Combined Arms Operations in Urban Terrain".* Washington, DC: GPO.

[9] Russell, S., Norvig, P. (2009). *"Artificial Intelligence: A Modern Approach."* (3rd ed.). Pearson Education.

[10] U.S. Army (2001). "*FM 3–90, Tactics".* Washington, DC: GPO.

[11] Boyd, John Richard (1976), *Destruction and Creation*, US Army Command and General Staff College.

[12] International Organization for Standardization (1989). *"ISO/IEC 7498-4:1989 -- Information technology-Open Systems Interconnection-Basic Reference Model"*. ISO Standards Maintenance Portal. ISO Central Secretariat.

[13] Hart, P. E., Nilsson, N. J., Raphael, B. (1968). *"A Formal Basis for the Heuristic Determination of Minimum Cost Paths"*. IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2): 100–107. doi:10.1109/TSSC.1968.300136

[14] U.S. Army (2007). *"FM 3–21.8, The Infantry Rifle Platoon and Squad*." Washington, DC: GPO.

[15] U.S. Army (2004). "*FM 101–5–1, Operational Terms and Graphics."* Washington, DC: GPO.