Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

SOFTWARE ENGINEERING A MULTI-LAYER AND SCALABLE AUTONOMOUS FORCES "A.I." FOR PROFESSIONAL MILITARY TRAINING

Michael J. Pelosi
Michael Scott Brown

ITEC Software Engineering Department
University of Maryland University College
Adelphi, MD, 20783, USA

ABSTRACT

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 and 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. Ideally, such an architecture 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 is desired. 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.

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 (Lane et al. 2005, Johnston et al. 2015).

The military simulation community has been working on similar proposed solutions for decades. A whole simulation conference series addressed issues relevant to this paper: the Computer Generated Forces/Human Behavior Representation (CGF/HBR) series, later renamed in Behavior Representation and Implementation M&S (BRIMS). A NATO Technical Report on Computer Generated Forces Forces/Human Behavior Representation (CGF/HBR) series, later renamed in Behavior Representation and Implementation M&S (BRIMS). A NATO Technical Report on Computer Generated Forces Technology (NATO Document Nr: RTO-TR-11 AC/323(SAS)TP/8) described similar solutions as desirable objectives in 1999. A chapter by Bharathy, Yilmaz, & Tolk (2012) on "Agent directed simulation for combat modeling and distributed simulation," in Engineering Principles of Combat Modeling and Distributed Simulation, gives several related examples and points to related research. Previous related Winter Simulation Conference papers include Cioppa et. al. (2004), Middleton (2010), Kuramoto & Furuichi (2013), Løvlid et. al. (2013), and one of the early papers presented at WSC which was revolutionary at the time is Karr & Franceschini (1994). The present paper is embedded into a rich WSC and SISO history acknowledged here.

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 (U. S. Army 2003). Software tools at the company, battalion, and brigade training levels have been sparse. One frequently utilized piece of software is the VBS simulation, which 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 largely nonexistent. Even the widely utilized VBS platform has a woefully inadequate AI — cooks and mechanics are routinely tasked to drive trucks, fly planes and helicopters, and play civilians, as part of the simulation exercise.

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 (Shannon 1950, Newborn 2012). 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.

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.

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 (see next page for descriptions). These processes can and have been modeled almost directly in code.

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."*

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.

Where possible the military decision-making process (MDMP) is then followed both in modeling and implementation. The COAI is designed to follow a process similar 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 (*T*roop *L*eading *P*rocedures) 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 in regard to slowing, diverting, or stopping movement.

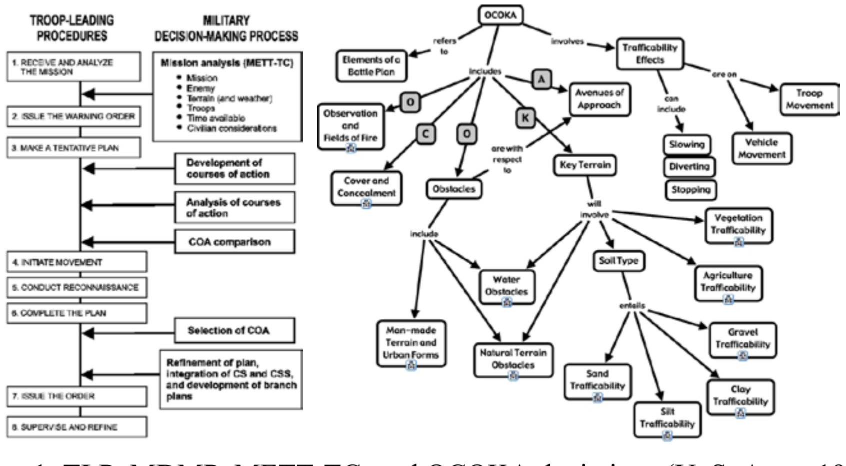

Figure 1: TLP, MDMP, METT-TC, and OCOKA depictions (U. S. Army 1997).

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.

Scenario design inputs to the COAI for information analysis include: time limit constraints, enemy order-of-battle (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 times 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 $\geq$ 3:1 points total advantage will be necessary for successfully taking the occupation of locational objective from a defender (U. S. Army 2002).
3. Analysis and calculation of a tentative plan to reach the goal end state. This is accomplished by using a brute force approach similar to the solving of the Traveling Sales Person problem (Russell and Norvig, 2009). 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.

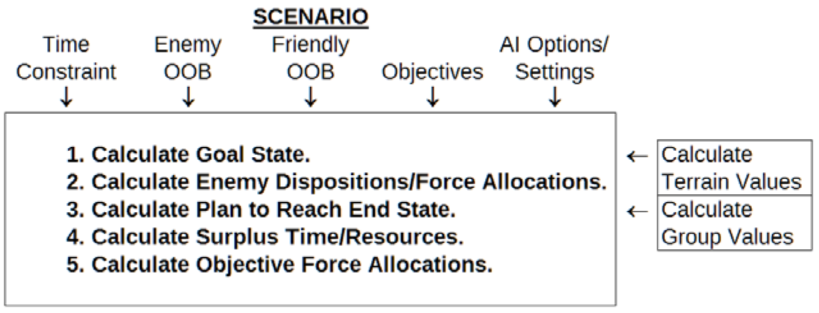
SCENARIO
Time Constraint
Enemy OOB
Friendly OOB
Objectives
AI Options/ Settings
1. Calculate Goal State.
2. Calculate Enemy Dispositions/Force Allocations.
3. Calculate Plan to Reach End State.
4. Calculate Surplus Time/Resources.
5. Calculate Objective Force Allocations.
Calculate Terrain Values
Calculate Group Values

Figure 2: Scenario and mission specifications and analysis.

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.

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 as shown in Table 1.

Table 1: Static and Dynamic Map 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 seen as 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.
• *Countermobility* – places where the enemy's movement can be stopped efficiently.
• *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.

As a consequence 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, OOB analysis is capable of assigning various groups to specific objective missions. For example, an armor company may only include 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.

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.

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, are able to 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 (see Russell and Norvig, 2009).

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 (U. S. Army 2001). 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.

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.

The execution phases are analogous to the very important military precept of the *OODA* loop (Boyd 1976). 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.

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. Transition to the end-game occurs 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, in order 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.

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, execution of reconnaissance and counter reconnaissance missions, and the seizing of any easy objectives closer than the enemy.

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 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* (U.S. Army 1997).

The endgame embarks on achieving the final scenario objectives. 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.

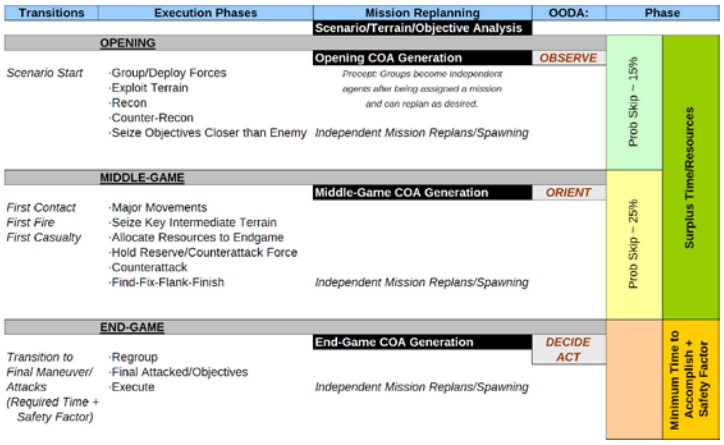
Transitions
Execution Phases
Mission Replanning
OODA:
Phase
Scenario/Terrain/Objective Analysis
OPENING
Opening COA Generation
OBSERVE
Scenario Start
·Group/Deploy Forces
·Exploit Terrain
·Recon
·Counter-Recon
·Seize Objectives Closer than Enemy
Precept: Groups become independent agents after being assigned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

Figure 3: COAI execution phases and major aspects of each phase.

Analogous to the IOS (International Organization for Standardization, 1989) model 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, and also 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 2: 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.

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 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.

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.

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.

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. Scripting can be manually controlled as desired, and saved to file.

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.

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.

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.

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 the 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.

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 React To Contact (RTC) events using an 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.

Pathfinding to mission waypoints along a movement route is calculated using a modified version of the A* Algorithm (Hart, Nilsson, and Raphael 1968). 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.

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.

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* (U. S. Army 2007). 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 those directions. 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. 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. 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 periodically relocate based on their proximity to the enemy. Headquarters units, for example, 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.

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.

As mentioned, COAs are implemented as a class object, and they 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.

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.

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.

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.

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.

REFERENCES

Bharathy, G. K., L. Yilmaz, and A. Tolk. 2012. "On Agent Directed Simulation for Combat Modeling and Distributed Simulation." *Engineering Principles of Combat Modeling and Distributed Simulation*, 669-713. John Wiley & Sons.

Boyd, J. R. 1976. "*Destruction and Creation*." US Army Command and General Staff College.

Cioppa, T. M., T. W. Lucas, and S. M. Sanchez. 2004. "Military Applications of Agent-based Simulations." In *Proceedings of the 2004 Winter Simulation Conference,* edited by R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 259-264. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Hart, P. E., N. J. Nilsson, B. Raphael. 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

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.

Johnston, J. H., G. Goodwin, J. Moss, R. Sottilare, S. Ososky, D. Cruz, and A. Graesser. 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 Ground, MD.

Karr, C. R., and R. W. Franceschini. 1994. "Status Report on the Integrated Eagle/BDS-D Project." In *Proceedings of the 1994 Conference on Winter Simulation*, edited by J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, 762-769. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kuramoto, K. and M. Furuichi. 2013. "Fuse: A Multi-agent Simulation Environment." In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, 3982-3983. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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

Løvlid, R. A., A. Alstad, O. M. Mevassvik, N. de Reus, H. Henderson, B. van der Vecht, and T. Luik. 2013. "Two Approaches to Developing a Multi-agent System for Battle Command Simulation." In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, 1491-1502. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Middleton, V. 2010. "Simulating Small Unit Military Operations with Agent-based Models of Complex Adaptive Systems." In *Proceedings of the 2010 Winter Simulation Conference,* edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 119–134. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Newborn, M. 2012. *Kasparov Versus Deep Blue: Computer Chess Comes of Age.* Springer Science & Business Media.

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

Shannon, C. 1950. "Programming a Computer for Playing Chess." *Philosophical Magazine* 41 (314)

U.S. Army. 1997. *FM 101–5, Staff Organization and Operations*. Washington, DC: GPO.

U.S. Army. 2001. *FM 3–90, Tactics*. Washington, DC: GPO.

U.S. Army. 2002. *FM 3–06.11, Combined Arms Operations in Urban Terrain*. Washington, DC: GPO.

U.S. Army. 2004. *FM 101–5–1, Operational Terms and Graphics*. Washington, DC: GPO.

U.S. Army. 2007. *FM 3–21.8, The Infantry Rifle Platoon and Squad*. Washington, DC: GPO.

Uwe K., J. Dompke, and G. Scheckeler. 1999. "Computer Generated Forces Technology." *NATO Report RTO-TR-11 AC/323(SAS)TP/8*, Research and Technology Organization, Neuillysur-Seine Cedex, France.

AUTHOR BIOGRAPHIES

MICHAEL J. PELOSI is Professor of Software Engineering at UMUC. He received his Ph.D. from Nova Southeastern University, Ft. Lauderdale, FL. His research interests include software engineering and artificial intelligence. His email address is michael.pelosi@faculty.umuc.edu.

MICHAEL SCOTT BROWN is Program Director of Software Engineering at UMUC. He received his Ph.D. from Nova Southeastern University, Ft. Lauderdale, FL. His research interests include software engineering and artificial intelligence. His email address is michael.brown@umuc.edu.