*Proceedings of the 1995 Winter Simulation Conference*
*ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman*

# SCALABILITY ISSUES IN ENHANCEMENT OF THE MAGTF TACTICAL WARFARE SIMULATION SYSTEM

Curtis L. Blais

VisiCom Laboratories, Inc.
10052 Mesa Ridge Court
San Diego, California 92121, U.S.A.

## ABSTRACT

The Marine Air Ground Task Force (MAGTF) Tactical Warfare Simulation (MTWS) system is a computer-assisted, two-sided warfare gaming system designed to support training of U. S. Marine Corps commanders and their staffs. Primary requirements for the system were written in the early 1980's. Since then, a transition in training from uni-service to joint and coalition warfare scenarios has occurred. Primary use of MTWS will continue to be within the Fleet Marine Force and USMC University settings. However, there are growing demands for the system to participate in joint exercises involving other constructive simulations and diverse virtual simulations. Therefore, the requirement to support exercises from Marine Expeditionary Unit (MEU) through Marine Expeditionary Force (MEF) levels is being extended to cover larger force structures with an order of magnitude increase in the number of game objects. The technical challenge is to significantly enlarge system capacity without sacrificing essential system performance and fidelity.

Several issues are being investigated to achieve this expanded capability. This paper describes hardware and software approaches and alternatives relating to architecture and functionality. For each alternative, the current capability is briefly presented, followed by a description and analysis of scalability issues and alternatives.

## 1 BACKGROUND

### 1.1 System Description

MTWS is a warfare gaming system designed to support training of U. S. Marine Corps commanders and their staffs. MTWS will primarily support Command Post Exercises (CPX) in which combat forces, supporting arms, and results of combat are modeled by the system. MTWS can be used to plan and rehearse tactical operations involving amphibious landings, air operations, fire schedules, and ground schemes of maneuver against a variety of opposing force operations under varying environmental conditions. The system can be operated at real-time, slower, or faster than real-time as directed by the system administrator.

### 1.2 System Architecture

#### 1.2.1 Hardware

MTWS executes on a distributed architecture consisting of one or more simulation processors, a system control workstation, and one or more user workstations. The system provides the flexibility to be configured to meet the size and needs of the supported exercise. The initial fielded configuration consists of three Hewlett Packard model 9000/755 workstations as simulation processors, one HP 9000/755 as the system control workstation, and twenty-six HP 9000/735 user workstations.

The simulation processors perform the combat models. From the system control workstation, the system administrator can allocate the combat models to one or more simulation processors. This is described more fully in the software overview to follow. The system control workstation manages the exercise clock, remote site communications, and data conversions between the simulation processors and the user workstations. The system control workstation has two Ethernet ports. One Ethernet connects the system control workstation with the simulation processors. The second Ethernet connects the system control workstation with the user workstations. This configuration is shown in Figure 1.

When operating in an Aggregate Level Simulation Protocol (ALSP) confederation, a separate processor is added to the simulation processor side of the configuration to connect to and communicate with the ALSP translator. This is shown in Figure 1 in dashed lines to indicate that this capability is optional.

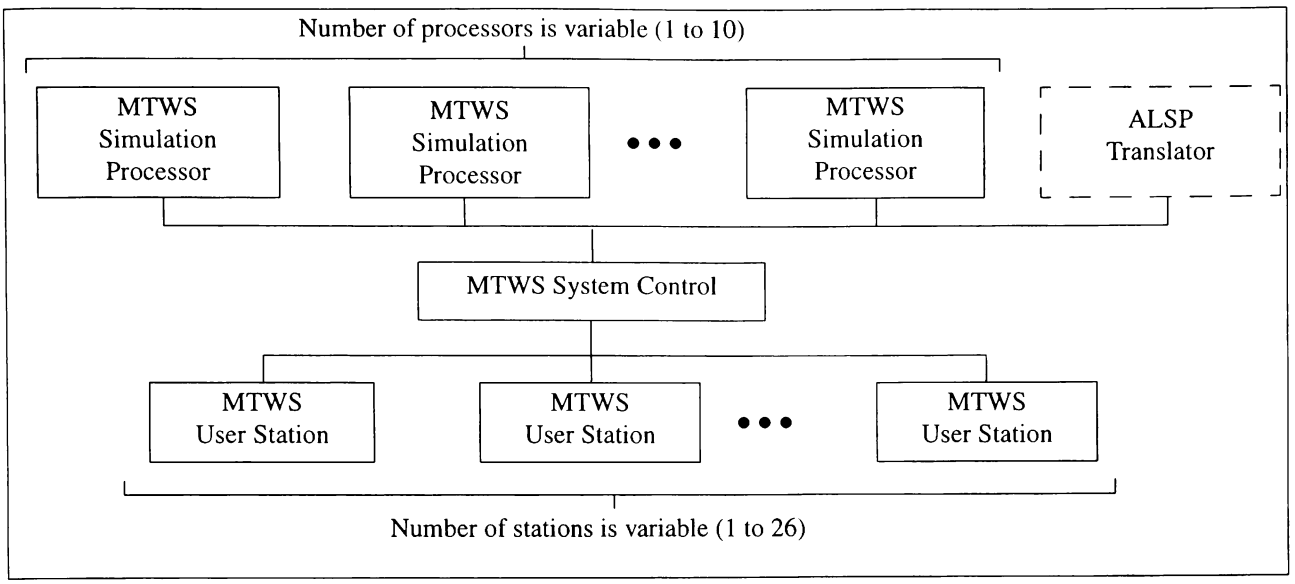


Figure 1: MTWS Hardware Configuration

### 1.2.2 Software

The MTWS software consists of three Computer Software Configuration Items (CSCIs), generally corresponding to the main hardware components described above. The CSCIs are identified as follows:

(1) MTWS Application Network (MAN), providing the combat models and executing on the simulation processors

(2) MTWS System Control (MSC), providing system operations, exercise control, clock control, data management, and report generation executing on the system control workstation

(3) MTWS Display System (MDS), providing command entry, spot report output (reporting the occurrence of simulated events), report request and display, map display, and tactical situation display executing on the user workstations

The system provides the capability for the system administrator to allocate MAN combat functions to one or more simulation processors. Allocable functions are Ground Combat, Ground Movement, Intelligence, Combat Service Support, Combat Engineering, Ship-to-Shore, Fire Support, Air Operations, Nuclear/Biological/Chemical (NBC) Warfare, and Weather. A single function cannot be divided across multiple processors, but multiple functions can be assigned to a single processor. Full distribution of functions, then, is assignment of each function to one of ten simulation processors. A sample allocation of functions across three simulation processors is shown in table 1. This scheme gives the system administrator the ability to balance the load across processors. Allocation decisions can be based on load conditions particular to the current exercise or on load conditions occurring at some point during an exercise.

Table 1: Sample Allocation of Combat Functions to Three Simulation Processors

| MTWS Simulation Processor | Function |
|---|---|
| MAN001 | Ground Combat<br>Ground Service Support |
| MAN002 | Ground Movement<br>Intelligence<br>Ship-to-Shore |
| MAN003 | Air Operations<br>Fire Support<br>Combat Engineering<br>NBC<br>Weather |

For example, if there are three simulation processors and the exercise conduct will have high air and ship-to-shore activity, then the Air Operations function could be assigned to one processor, the Ship-to-Shore function to a second, and all other functions to the third processor. As the operation transitions to heavier land battle, with significant reduction in ship-to-shore activity, the Ground Combat and Ground Movement functions, for example, could be moved from the third processor to the second to make better use of computing resources. The reallocation is strictly a manual operation–there is currently no automatic load balancing across the simulation processors. It is also interesting to note that the same executable program is loaded and started in each simulation processor. The processor receives a message from the system

control workstation indicating which functionality to activate. The system control workstation and the simulation processors communicate data base changes through messages over the Ethernet. Each simulation processor holds a portion of the exercise data base needed to perform its assigned functions. The system control workstation maintains a copy of all data from the updates received from the simulation processors.

Casualty/Damage Assessment (CA) cannot be allocated directly by the system administrator. Rather, CA is performed on each simulation processor hosting an associated combat function, as shown in table 2.

Table 2: Association of Casualty/Damage Assessment Capability with Simulation Processors

| Functional Area | Associated CA |
|---|---|
| Ground Combat | Ground Direct Fire; Organic Indirect Fire |
| Air Operations | Air -to-Surface; Air-to-Air; Surface-to-Air |
| Fire Support | Surface-to-Surface Indirect Fire |
| Ground Movement | Ground Minefield/ Barrier Contact |
| Ship-to-Shore | Ship Minefield/Barrier Contact; Ship-to-Shore Minefield/ Barrier Contact |
| NBC | NBC Contact |

### 1.2.3 Game Clock

Management of the game clock is not specifically addressed as a scalability issue in this paper, but is important to discuss in passing to achieve fuller understanding of the MTWS system. There are three main modes of clock management: (1) ALSP confederation control; (2) real-time synchronous; (3) event synchronous.

When operating as part of an ALSP confederation, the MTWS game clock generally runs in time constrained and time regulating mode (MITRE 1994). When time constrained, the MTWS clock is prevented from advancing time ahead of the confederation. When time regulating, MTWS participates in the advance of the ALSP time by requesting a time advance. The ALSP time will not exceed any time-regulating system's last requested time plus a lookahead value.

Whether the system is running as a member of an ALSP confederation or not, the system administrator can direct the system to run faster or slower than real-time. Of course, when operating in a confederation, the actual rate of time advance will depend on the time control selections described above. When operating independently, or when time constraint and regulation are disabled, MTWS time will advance in one of two modes, selectable by the system administrator. These modes are denoted real-time synchronous and event synchronous.

In real-time synchronous mode, the MTWS game clock advances at the same rate as the real-time clock ("wall clock"). If the administrator has directed the system to run at 2:1 advance rate in real-time synchronous mode, then the game clock advances one minute every 30 seconds of real-time. The MTWS system control workstation maintains the game clock, and notifies the simulation processors of the current game time every 15 seconds (real-time). This is an effective way of advancing time rapidly during periods of low activity, when so desired by the exercise control staff.

In event synchronous mode, the MTWS system control workstation does not advance the time until notified by the simulation processors that they have completed processing of all scheduled events for the previous time notice. For example, if the simulation processor performing the Fire Support function has a large number of fire missions scheduled for 0830, then the MTWS system control workstation will not be able to advance the game clock by 15 seconds (assuming the game is set at 1:1 time advance) until notified by this simulation processor that it has completed all events that were scheduled for 0830 (and 15 seconds of real time have elapsed). Running in event synchronous mode prevents scheduled simulation events from falling behind the game clock.

## 2 PROBLEM STATEMENT

There is growing demand in the Marine Corps command and control training community for representing large numbers of joint and coalition forces. Recent world events have demonstrated the benefit of joint response to crisis situations. The ever-increasing employment of the ALSP confederation of models is indicative of the growing interest in applying simulation and modeling to joint training exercises. To support large, joint exercises, the MTWS system must increase in capacity of game objects by an order of magnitude (10-30 times), without sacrificing essential system performance and fidelity. The MTWS specifications require representation of up to 600 surface objects (ground units and ships), 240 air missions, and 140 ship-to-shore elements, for a total of nearly 1000 game objects. In order to fully "ghost" objects from other members of the ALSP confederation, MTWS will need to accommodate quantities in the range of 12000 ground units, 1000 ships, and 2000 air missions.

As MTWS transitions from development to fielding in June 1995, the Marine Corps is interested in mapping out a growth path to increase system capacity. This raises a number of questions. To what degree was the MTWS system designed for scalability on this order? To scale up to this level, what sacrifices in model fidelity need to be made? How can the current investment in hardware and software be preserved, yet provide a foundation for growth on this scale? What hardware and software changes will be necessary to achieve this growth? In this paper, we provide a starting point for planning the growth of the MTWS system to achieve these goals.

## 3 ARCHITECTURAL SCALABILITY ISSUES AND APPROACHES

We need to first reach some agreement on the meaning of scalability with respect to MTWS considerations. From a hardware perspective, "scalability" generally indicates that the addition of hardware to a system results in greater problem-solving ability, and the ratio of hardware to system capability stays nearly constant as hardware is added. For example, if a system consisting of 2 processors is scalable, then expansion to 4 processors will result in nearly twice the original capability. "Capability" here may be represented as "processing speed"--the expanded system performs the same job twice as fast--or as "problem size"--the expanded system performs twice the original sized problem in approximately the same time. For problems that can be subdivided to achieve perfect parallelism, the increased capacity will be exactly twice the original configuration. In practice, adding processors creates overhead that reduces the gain in performance. If doubling the number of processors results in only a 50% improvement in overall capability, it may be argued that the system is not scalable. If the relative increase in capability continues to decrease as more processors are added, there may be even stronger evidence that the system is not scalable. Unfortunately, there are no clear criteria for making this judgment (Nussbaum and Agarwal 1992).

Although lacking in mathematical rigor, we take as our basis the notion that "a scalable architecture exhibits speedup linearly proportional to p, the number of processors employed" (ibid.). If the problem size remains constant, then a scalable architecture will exhibit speedup linearly proportional to the number of processors employed. If the problem size increases, then for a proportional increase in the system, a scalable architecture will maintain performance roughly equivalent to that attained by its original configuration for the original size problem. We use the term "architecture" rather than just "hardware" to emphasize that a scalable solution will often involve both hardware and software.

We identify below several architectural options for increasing the capacity of the system, while preserving essential functionality, performance, and hardware/software investment. No single option will provide the full solution to scalability. Rather, the cumulative effect of one or more of these options, together with algorithmic changes discussed later, will enable the system to make rapid progress toward the enhancement goals.

### 3.1 Multiple MTWS Confederation

One interesting side-effect of the integration of MTWS into an ALSP confederation is the capability to tie together multiple MTWS suites to conduct a single exercise. For example, the I MEF system in Camp Pendleton, California, could communicate with the II MEF system in Camp Lejeune, North Carolina, to conduct a two-MEF exercise. Although data base quantities must be effectively doubled to accommodate all owned and ghosted entities, the processing load would be shared by both suites of equipment. Software logic could localize the additional processing load to situations involving an overlap in the tactical situation being modeled; for example, where ground units from one system are interacting with ghosted ground units from the other system.

Efforts are underway in the ALSP community to define ground-to-ground interactions for interoperability between MTWS and the U. S. Army's Corps Battle Simulation. The resulting capability will improve interoperability among multiple MTWS suites.

### 3.2 Hardware Upgrade

The MTWS development has enjoyed the benefit of the rapid increase in hardware technology over the past five years. The initial development platforms were the MILTOPE militarized HP 9000/350 series machines purchased from the Army Common Hardware contract in 1990. These machines were rated at 8 million instructions per second (8 mips). The project has since transitioned through the HP 9000/400 series to the HP 9000/700 series machines purchased from the Navy TAC-3 contract. As described earlier, the initial fielded systems (Camp Pendleton, Camp Lejeune, Okinawa, and Quantico) use the HP 9000/735 machines for the user workstations, and HP 9000/755 machines for the simulation processors and system control workstation. These machines are rated at 120 mips. This last upgrade was made in December 1994 and provided an overall improvement in processing time performance of 25-30% over the predecessor 730 and 750 machines. Processing speed improvement translates into a capability to solve larger problems within established performance bounds.

As the hardware evolved upward, higher speed and greater memory densities became available, at approximately the same price as the earlier hardware. Because the USMC delayed procurement decision until late in the development, it was able to take advantage of this evolution. As the USMC looks ahead to future fielding (e.g., Twenty-Nine Palms, California, and Marine Reserves in New Orleans, Louisiana), it can opt to buy the best available at the time of purchase. Adherence to open systems architecture and industry standards (such as Unix, X/Motif, Ada, Ethernet, TCP/IP) has made the upward progression largely effortless. Continuing this trend, the recent award of the TAC-4 contract to Hewlett Packard gives the system a direct growth path to the HP 9000/770 machines, an architecture offering up to two processors, rated at 150 mips each, in a single cabinet, with 1 Gigabyte of memory (4 times current workstation capacity). The cost of a dual processor HP 9000/770 with full memory is less than the cost of two HP 9000/755 machines from the TAC-3 contract.

In parallel with USMC development efforts, VisiCom Laboratories has initiated an Internal Research and Development effort to host MTWS software on a VME-based, multiprocessor system. VME (VERSAmodule Europe) is an international, commercial standard for board-level communications (IEEE 1987). The initial porting effort was completed in October 1994 and successfully demonstrated integration of 4 processor cards in a single card cage, duplicating the functionality of the system control workstation and 3 simulation processors. Each processor is equivalent to that in an HP 9000/750 system. This approach offers significant reduction in hardware size and cost, with opportunity for increase in the number of simulation processors in the system. A future focus of the effort is to move from interprocessor communication over Ethernet to shared memory accessed across the VME system backplane. When this capability is implemented, MTWS performance benchmark tests will be run to provide comparison of this architecture to the fielded version of the system executing on HP 9000/755 workstations.

### 3.3 Function and Object Allocation

Another option for increasing system capacity is to enable greater separation of functionality to allow allocation to a larger number of processors. For example, the system could be modified to separate the air defense functionality from air operations so they could be assigned to separate processors when the air warfare portion of the exercise is very large or when operating in a large, joint wargame. Additionally, new or expanded features can be made allocable as they are implemented. As functionality assigned to each simulation processor is reduced (i.e., spread over other processors), the capacity of that processor for the remaining functionality is increased. The trade-off that must be controlled is increased interprocessor communications traffic for coordinating simulation events and transfer of data base updates.

CA calculations can be very computationally intensive when there are large numbers of game objects or when there are large quantities of simulation events invoking CA. Another functional separation option is to move CA processing from the associated combat model functionality and perform it on one or more processors dedicated to CA calculations. Specific types of CA could be performed on their own dedicated processors, or a CA "server" could be designed to accept any request for assessment and return the assessment results.

A relatively simple enhancement to the allocation scheme may provide the most dramatic improvement. Instead of distributing functionality over a set of processors, add an option allowing distribution of responsibility for particular functionality AND a specific set of game objects to separate processors. For example, allow the user to split ground visual detection processing across two processors, each responsible for a user-specified portion of the game area or subset of game objects. Analysis of this option will attempt to identify partitions that minimize interprocessor communications that could otherwise undermine the processing speed and capacity gains.

## 4 ALGORITHMIC SCALABILITY ISSUES AND APPROACHES

Algorithmic scalability is exhibited when the processing time and storage use of an algorithm increase linearly proportional to the increase in the problem size. Best case is achieved by algorithms that accomplish, for larger problem sizes, the same functionality in less time or equal time than the original algorithm.

Consider a simple example based on the original MTWS design and implementation for determining visual detection between Landing Force (LF) and Opposing Force (OPFOR) units. The algorithm considered each LF unit and made a detection determination against each of the OPFOR units based on such factors as distance, line-of-sight, vegetation cover, and visibility conditions. This process was then reversed, with OPFOR units attempting to detect LF units. This simplistic approach is clearly not scalable by the above definition. If there are an equal number of LF and OPFOR units, N, then there were 2*N**2 ("N-squared") determinations made. If the number of units doubled to 2N, then the processing time (effectively) increased by a factor of 4. An alternative, scalable algorithm should do little worse than double in processing time when the problem size -- in this case, the

number of units -- doubles. We have since implemented an algorithm that exhibits such scalability, as will be discussed later.

There are a number of basic strategies for increasing speed of algorithms as the problem size increases:

(1) Replacement. The algorithm can be replaced with one designed for scalability.
(2) Simplification. The algorithm can be simplified to perform less processing, but still meet the established functional requirements.
(3) Relaxation. Model detail, specified in the software requirements, can be relaxed and the algorithms modified or replaced to provide the reduced capability.
(4) Aggregation. The data objects can be grouped to form aggregated entities, and the algorithm can be modified or replaced to deal with the aggregated objects.
(5) Selection. Multiple levels of model detail can be added, possibly offering user-selected algorithms derived from the previous four approaches.

We describe below examples of these approaches as they relate to several functional areas of the MTWS model. The functions deal with three aspects of ground combat modeling: visual detections, casualty/damage assessments, and ground unit representation. The processing performance of these functions is directly impacted by increasing the number of ground objects in the system. Other model areas require similar analysis, but are beyond the scope of this paper.

### 4.1 Ground Visual Detection Modeling (Replacement)

We looked at the original MTWS visual detection algorithm as outlined above. To move toward a more scalable solution, we adapted an algorithm from the Distributed Interactive Simulation (DIS) program which we call the "proximity algorithm." Briefly, the algorithm places objects in "proximity bins." Each bin is a cubic region in three-dimensional space. The bins are disjoint. Ground units (or any other game object) are placed into a bin by a hashing algorithm on the unit's location. The length of the side of each bin is chosen by the application developer based on the needs of the application. The maximum visual detection range in MTWS is set at 8 kilometers (km). Therefore, the length of a bin side was chosen to be 16 km. As described earlier, the original detection algorithm considered all other-sided units to determine what units could be detected. With the proximity algorithm, the logic only considers candidates in the bin containing the unit, and bins adjacent to the quadrant in which the unit lies. For illustration purposes, a two-dimensional cross-section is shown in Figure 2 (for ground objects, bins above and below the one containing the unit in question seldom contain other ground objects for consideration). In general, only four bins need to be used to determine if there are other-side units that can be detected.

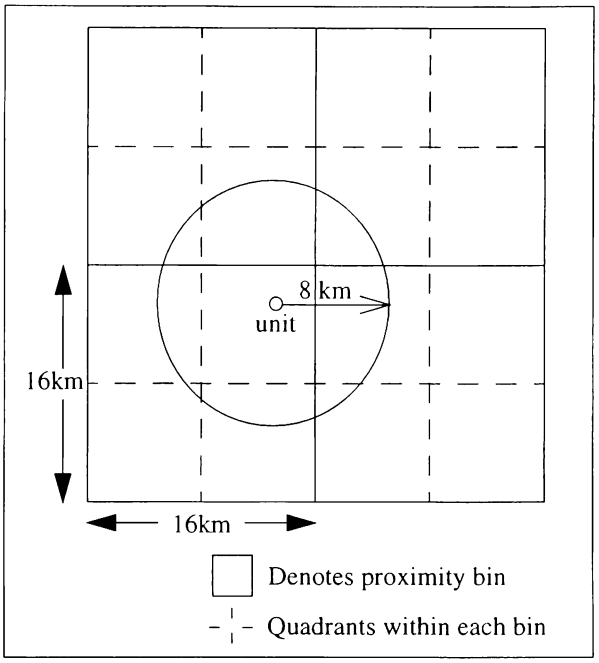


Figure 2: Identification of Proximity Bins Containing Candidate Units for Visual Detection

As a further refinement to the proximity algorithm, bins are identified by "side" (LF, OPFOR, civilian, none, multiple) based on the ground units contained in the bin. This limits detection processing to only those bins containing units from other sides or having adjacent bins with units from other sides. In normal tactical situations, many ground units represented in an exercise are rear echelon elements (e.g., combat service support, reserve maneuver element) that will seldom come within visual detection range of other forces.

The new algorithm exhibits the desired scalability behavior, which can be demonstrated as follows. Suppose in a two-sided game that units are heavily interspersed. That is, suppose there are B bins containing N units, with N/2 units on each side, and N/2B units of each side within each bin. Using the proximity algorithm under these conditions, the processing time, P, for determining detections is approximated by:

$$P = kB(N/2B)(4N/2B)$$

where

k is a constant (contains the factor of 2 for each side considering the other),
B is the number of bins containing units,
N/2B is the number of units on one side in a bin,

4N/2B is the number of units from the other side that need to be considered.

Grouping the constants and simplifying, we get:

$$P = k(N/B)N = kDN$$

where we call D the average density of the exercise (i.e., average units per bin).

We contend that D is bounded, determined by the level of representation of ground units within the exercise. If so, then this approach achieves linearity in growth of N that we desired. There are a number of dynamics here that need to be considered. Exercises are typically constructed for a desired level of play based on training objectives and the staff to be trained. Design of the exercise data base establishes the ground unit representation level appropriate to the exercise. For example, a MEU exercise to train the battalion commander and staff may have units represented down to the platoon or even fire team level. A MEF-level exercise, may have units represented down to the company level. In ALSP exercises (e.g., Ulchi Focus Lens 1994), ground forces in a theatre-level exercise have been represented down to the company level. Regardless of the exercise set-up, the selection of representation level for that exercise results in an implicit value for the density, D. Ground forces occupy space, defined by front and depth dimensions, and have a tactical spacing between them. Therefore, for any given level of play, there will be a limit to the number of ground forces that will occupy a particular area. If the number of units at a given representation level are increased, then they will occupy a larger number of bins. If the representation level is increased, then the density will decrease; for example, the density for battalion-sized units will be less than the density for company-sized units.

It was noted in (Steinman and Wieland 1994) that "military simulations tend to behave like football games where everyone chases the player with the ball... most of the ground units would inevitably congest into a small number of grids to fight their battle. Because grids performed most of the work, they became bottlenecks, thus limiting the amount of parallelism in the simulation." This is true up to a point. Certainly in a ground battle, units will gather for assaults, defense, or counter-attacks. However, for a given level of representation (company, battalion, etc.) the density of units within a proximity bin would not exceed a maximum value for that level, at least not without potentially disastrous combat results. The inextricable trend in warfare is toward increasing dispersion as range, accuracy, and lethality of weapons increase.

### 4.2 Casualty/Damage Assessment Modeling (Simplification, Selection)

CA calculations in MTWS occur at the individual target asset level; that is, the tanks, trucks, anti-tank weapons, and other items possessed by a unit receiving fire. The proximity algorithm described above has already been implemented to identify units within the target area that could be affected by the fire. For example, if 36 rounds of artillery fire are delivered on a target location, the impact point for each projectile is computed. The distance from the target location to the furthest impact point is used as the radius of a circular area within which ground units (and other objects, such as ship-to-shore elements and bridges) may be subject to damage. The proximity algorithm is used to reduce the candidates to those objects in bins that intersect the circular area. Individual target assets are randomly located within the area covered by each candidate ground unit. The distance from each target asset to each impact point and the mean area of effectiveness (MAE) of the ordnance are used to compute a cumulative probability of damage, from which it is determined if the asset is damaged or destroyed.

This level of detail may be appropriate for small numbers of units, or even for large exercises used for analytical purposes, but could be simplified significantly to achieve higher performance. One approach would be to consider the unit location as the average target asset location, and compute the cumulative probability based on that location and the distance to the impact points. Or, use the unit location and a location computed as the average of the impact points. The computed probability could be used as the mean of a normal distribution, with a standard deviation definable by the user in parametric data or determined from the accuracy of the firing weapon. For larger exercises, the detail in the assessment algorithm may not be as critical, as long as reasonable results are obtained.

Alternative CA algorithms can be made available in the system for selection by the user based on the objectives of the particular exercise. Such selections could be offered prior to start of the exercise, or even during exercise conduct, to enable the user to determine the level of fidelity desired.

### 4.3 Ground Combat (Relaxation, Aggregation)

To obtain spatial distribution of assets across the area occupied by a unit, MTWS represents ground units using 3 circular subelements. Subelement positions are based on unit location, formation (line, echelon left, vee, etc.), and unit front. All of these attributes have default values set in the model parametric data and are modifiable by the user. Recent discussions with the USMC requirements

community indicate that this level of detail is seldom, if ever, needed by the system users. By relaxing requirements that call for this representation of ground units, processing time can be reduced in ground visual detection, CA, and ground combat functions, with added savings in data storage. Formations would still come into consideration for visual detection processing and CA calculations, but would be represented by appropriate shapes (rectangle, triangle, and circle) to determine the area occupied. The challenge would be to design algorithms using these shapes that would reduce the processing time compared to the current handling of multiple subelements. Although elimination of subelement representation is a modification to the software requirements, such action would not degrade essential model fidelity or capability.

The current implementation creates separate asset records for each component of a "complex" equipment item. For example, an M1A1 tank consists of a platform (chassis), a main gun, and two machine guns. All four components are stored as separate assets of the unit possessing this asset. Representation at this level allows the possibility for individual components of an equipment item to be in different states (operational, damaged, destroyed). As currently implemented, the system considers the susceptibility of the components to damage, but sets the status of the item of equipment as a whole. If the M1A1 tank chassis suffers a mobility kill, then the item as a whole is considered to be in a mobility kill status. If, in the current system or when larger exercises are run, it is acceptable for the software to continue to work at the aggregated equipment level, then the creation of separate asset records for the components is not necessary. Elimination of separate asset records for equipment components will provide improved performance in all areas in the system where unit asset records are processed (e.g., movement, ground detection, CA), and will significantly reduce storage of asset data for units in the exercise. This change would not remove consideration of component capabilities for deciding which weapon to fire, just the tracking of component quantities within the assets possessed by a ground unit. Furthermore, the decision to run the system with or without the component level of representation can be made into a user-selectable feature.

## 5 CONCLUSION

The MTWS project faces an exciting growth challenge. To participate effectively in large, joint training exercises, MTWS must grow in quantities of exercise objects by as much as 30 times without degrading system performance or fidelity. Several ideas and approaches have been presented in this paper that begin to move the system toward this goal. These approaches provide a starting point for prototyping and performance comparison efforts. No single method described is likely to provide the full answer. Rather, judicious selection of a combination of methods providing the greatest short-term improvements will be the most cost-effective solution to the Marine Corps.

The fundamental architecture of MTWS is proving to be an excellent foundation for building new and added capabilities. It has already demonstrated its flexibility in the addition of ALSP capabilities and adaptability to evolving hardware products.

## ACKNOWLEDGEMENTS

Thanks to Jason Zions, Network System Management Consultant and SYSOP, CompuServe HP Systems Forum, for his perspective on architectural scalability.

The views expressed in this paper are those of the author and VisiCom Laboratories, Incorporated, and do not necessarily express the official position or policies of the United States Marine Corps.

## REFERENCES

MITRE Corporation 1994. *Aggregate Level Simulation Protocol System Software Version 7.0 User Manual*, Informal Report.

IEEE Standard 1014, VMEbus Specification 1987.

Nussbaum, D., and Agarwal, A. 1992. Scalability of Parallel Machines. *Communications ACM* 34, 3 (March 1992), 57-61.

Steinman, J., and Wieland, F. 1994. Parallel Proximity Detection and the Distribution List Algorithm. *Proceedings of ELECSIM 1994 Electronic Conference on Constructive Training Simulation.*

## AUTHOR BIOGRAPHY

**CURTIS L. BLAIS** is Manager of Wargaming Systems for VisiCom Laboratories, Inc., and Software Engineering Manager for the MTWS project. He has twenty-one years of experience in analysis and simulation of Navy and Marine Corps command and control systems and communications networks, and in design and development of combat models. He specializes in modeling of ground combat and casualty/damage assessments. Mr. Blais holds B.S. and M.S. degrees in Mathematics from the University of Notre Dame.