TECHNICAL ARTICLE

*SIMULATION* 74:3, 138-161

ISSN 0037-5497/00
Printed in the United States of America

# Simulations Technologies in the Mission Operational Environment

**John R. Surdu** and **Udo W. Pooch**
Texas A&M University
Department of Computer Science
College Station, Texas, USA
E-mail: {surdu | pooch}@cs.tamu.edu

*This paper proposes a methodology for applying simulation technology to the monitoring and control of operations, focusing on the use of intelligent agents. These agents monitor the course of the real operation and compare it with the near-real-time simulation of that operation. When these agents detect significant differences between the planned operation and the real operation, they explore the discrepancies to determine the impact on the desired outcome of the operation. These agents can launch additional simulations or other tools to make this determination. In cases in which the success of the plan is threatened, the agents advise the decision maker. The contributions of this research are (1) to help identify the requirements for a simulation designed for use during operations and (2) the development of a methodology for using simulation, along with rational agents, to compare the real, ongoing operation with the planned operation and make recommendations when appropriate. While several training simulations have been developed, there are currently no tools that provide this simulation-based, agent-assisted operation monitoring/ analysis capability to decision makers.*

**Keywords**: Operation monitoring, software agents, command and control, military simulation

## 1. Introduction

A large number of tasks performed by commanders and staffs can be facilitated during operations by the application of simulation technologies. Traditionally the focus of simulation in the Department of Defense (DoD) has been on analysis and training. Simulations designed to facilitate course of action (COA) development and analysis, rehearsal, and operations monitoring will enhance the effectiveness of staffs and commanders. Currently, there are no *operationally-focused* simulations, those built specifically for use during operations.

The Army Modeling and Simulation Office (AMSO) recognized the importance of simulation in command and control, and they identified five voids in current modeling and simulation technology for the Army After Next[1] [1]. According to their analysis, the first of these voids, Cognitive Modeling, includes automated decision aids, COA tools, and tactical information aids. The methodology proposed in this paper, originally described by Surdu and Pooch [2], intends to address these three technology voids. The Defense Advanced Projects Research Agency (DARPA) has also recognized the importance of simulation in command

[1] Army After Next is the vision of the structure and doctrine of the U.S. Army after 2010. It is an ongoing process.

and control activities. A DARPA concept briefing for the Command Post of the Future (CPoF) project provides a list of several tools that will provide input to the *Battlespace Reasoning Manager*. Among these are "Planning and Analysis Applications" and "3D Models and Simulations." In another portion of the briefing, DARPA notes that "Battlespace Reasoning, Analysis, and Simulation" assist the commander's perception and understanding of the battlespace [3].

Clausewitz, a noted military strategist, in *On War (*particularly in his discussion of *friction)*, talks about the *feel of the battlefield* and how great commanders have this ability to deal with friction and see through the fog of war [4]. He also notes that this feel of the battlefield only comes with experience. Unfortunately, this experience must be gained at the cost of expensive maneuver exercises or ultimately in human life. The Army developed a number of facilities (such as the Combined Arms Training Centers (CTCs)) and training simulations (such as CBS, BBS, JANUS [5], ModSAF [6], JTS [7], and WarSim [8]), which attempt to build this experience at relatively low cost. In days of constrained budgets and 100-hour wars, the Army is short of ways of effectively identifying those officers who have this feel of the battlefield.

Surdu, Haines, and Pooch [9] discussed the uses of simulation for COA development and analysis and for conducting rehearsals. This paper elaborates these uses for simulation, and discusses the use of simulation during an ongoing operation as a tool to help the commander and his staff track the progress of the operation and anticipate problems. This paper proposes a methodology in which simulation technologies support commanders and staffs during actual operations. Finally this paper discusses the technical issues arising from this methodology.

## 2. Uses of Simulation During Operations

While the various military services and civilian organizations have different command and control processes, there is a common thread among them: planning, rehearsal, execution, and after-action review. Simulation technology can be applied in each of these phases.

**Planning:** During the planning phase, staffs develop courses of action (COAs). For instance, one COA might be a heavy attack on the left with a small unit delaying on the right. The current method, as outlined in FM 101-5, is an *ad hoc* process involving members of the staff discussing the various COAs. Each phase of the operation is analyzed according to *an action-reaction-counteraction* paradigm. This ad hoc method has numerous problems.

The effectiveness of an *action-reaction-counteraction* analysis of COAs is also dependent to a large extent on the interaction between the various members of the planning staff. The reality of the current personnel management policies is that a staff rarely has time to coalesce. Except during preparations for deployments to the large-scale training exercises, personnel rotations ensure that a fair portion of a planning staff will be new to the group.[2]

The same officers who develop the COAs are the ones who analyze them for strengths and weaknesses and determine the criteria used to evaluate the COAs. Despite the best intentions, the planning staff carries with it personal biases as to which plan is better than other plans. This notion of the developers also being the evaluators can lead to *group-think* [10], in which the decision developed by the group is unduly affected by a desire to conform. Given a bias toward one COA, it is easy to manipulate the criteria, weights on the various criteria, and resultant decision support matrix to support the pre-ordained "best" COA. This bias may be manifested consciously or unconsciously, but it is clearly a risk associated with this ad hoc procedure. In the current planning process, once the formal decision briefing to the commander commences, no one in the staff is likely to openly oppose the staff's COA recommendation. Normally, this group-think can be countered only by a forceful commander, assistant commander, or chief of staff.

With an operationally-focused simulation, the staff enters the enemy and friendly courses of action and then simulates them. The results of numerous simulation experiments provide feedback for the decision maker in choosing one COA over the others. The proposed simulation provides much better feedback to the planners about synchronization problems than the *current* ad hoc procedure. It is true that the use of a simulation is not a panacea. The parameters used to initialize the simulation can be biased. The attrition model can be inaccurate. The current implementation allows for the easy replacement of the attrition model with any other valid model. Additionally, the adaptive nature of the proposed system will help ensure that over time the simulation's parameters will tend toward "reality." (This adaptive process is described later in the paper.)

The staff can still propose "straw man" plans, plans that are intentionally sub-optimal. A staff usually proposes two valid courses of action and one "throw-away," since the commander often wants three choices; therefore, the staff only considers two viable COAs. This is due to time constraints; there is generally insufficient time to adequately analyze three COAs.

Our hypothesis is that a staff, armed with a valid simulation with which to conduct COA analysis, will be able to adequately analyze more viable COAs—and do a better job of analyzing the COAs—than under the current, manual, ad hoc method. *While the manual*

[2] The purpose of this paper is not to attack personnel management policies. This discussion is meant to describe one effect of current policies despite the advantages of those policies.

*method was appropriate in an industrial-age Army, it is no longer appropriate for an information-age Army that needs to make decisions faster than the enemy does.*

An additional advantage of a simulation-based process is that the decision maker can conduct experiments in parallel with his planning staff. Later in this paper, requirements for the operationally-focused simulation are described. One of these is that it be capable of being operated by a single user on a single workstation. Clearly, then, the decision maker can experiment with one or more COAs while his planning staff works on the same ones or others.

If time permits during military operations, the planning staff explores possible alternative actions during the operation (branches) and follow-on operations (sequels). Simulation of the plan makes it much easier for the decision maker and planning staff to explore such branches and sequels. With the operationally-focused simulation, these branches and sequels can be quickly simulated to provide feedback to the planners.

Finally, having the operationally-focused simulation at multiple echelons will speed the planning cycle. Military planning involves the top-down decomposition of plans and the decentralized, bottom-up execution of plans. In this process, subordinate commands cannot begin detailed planning until the bulk of the planning at the higher headquarters has been completed. Once a Division headquarters has completed the plan, they could transmit the plan file electronically to each of the subordinate Brigades. The Brigade planning staff can then cull out entities that are unlikely to affect them, partially disaggregate the entities in the Division plan to be appropriate at brigade level, and begin to flesh out the Brigade plan.

It is important to remember that this methodology is not designed to create courses of action. Fiebig, Hayes, and Schlabak have done some interesting work in generating courses of action in the military domain using genetic algorithms [11, 12]. The methodology proposed in this paper is designed to *assist* humans in planning, but it does not do any planning itself. It is not the job of this system to generate courses of action but to assess their results through simulation (and some artificial intelligence tools) and to advise the commander and staff when the probability of success is low. Because this system does not do the actual planning, many of the problems associated with AI planning systems are not a factor (e.g., they work well in some domains and situations, but not others; early commitment to the ordering of subtasks that requires replanning; elimination of redundant steps; and conflict resolution arising from competing goals or subgoals). Humans do the planning.

**Rehearsal:** Once a COA has been chosen, it is developed into a full plan and that plan is rehearsed. The real purpose of a rehearsal is to identify synchronization issues and to make sure that everyone fully understands the plan. Certain rehearsals (e.g., fire support rehearsal) are difficult to conduct over sand tables and maps. Clearly simulation would be an asset for these types of rehearsals; however, a simulation-based rehearsal would also be useful for the traditional, maneuver-centric rehearsal. A simulation that can be halted at will could facilitate a rehearsal just as large sand tables and map boards do today.

A significant advantage of a simulation-based rehearsal is that it could potentially be distributed geographically. With a number of distributed graphical interfaces connected to the same simulation, the commander and operations officer could control the execution of the playback of the plan while the subordinate commanders and other staff members watched at remote locations. The rehearsal could be conducted without all the key players getting within grenade burst radius of each other.

**Execution:** After the plan has been chosen, refined, and rehearsed, and the operation commences, the proposed methodology can be used to monitor the progress of the simulated plan and the real operation. Operations Monitors compare the progress of the real plan against the simulation of that plan. When significant deviations from the plan occur, the Operations Monitors launch tools that explore the impact of the deviations. Finally the commander is advised if the Operations Monitors determine that the success of the plan is in jeopardy. The remainder of this paper focuses on the use of simulation during the execution of an operation.

**After-Action Reviews:** After-action reviews are important—even during a war. The course of the real operation could be recorded and archived for later review. As time permits, the operation could be "played back" for the key leaders. This would give commanders and staffs the opportunity to identify synchronization problems or other errors that lead to the final outcome of the operations. During training exercises there are often observer/controllers to dispassionately observe the conduct of planning and operations and provide feedback afterwards. This capability is unlikely to be available during real operations. The use of an operationally-focused simulation could help fill this void.

## 3. Unsuitability of Training Simulations

The military community has developed a large number of simulations for training and analysis, such as the Corps Battle Simulation (CBS), Brigade and Battalion Battle Simulation (BBS), JANUS, and Modular Semi-Automated Forces (ModSAF). While many of these are excellent products, most are unsuitable for use during an operation for a number of reasons, including large pre-exercise preparation, specialized hardware, large numbers of required participants, and large numbers of required workstations.

Surdu, Haines, and Pooch [13] enumerated the desirable capabilities for an operationally focused simulation to be used during operations, and they include:

- The simulation must be runnable from a single workstation by a single user. During ongoing operations, operations centers are crowded, bandwidth is limited, and contractor support is limited. A simulation that cannot be run by a single person on a single workstation would represent a significant burden to an already busy staff. This requirement was recently validated by the Force XXI Project's search for a "low-overhead" training exercise driver (simulation) and by Blais and Garrabrants in a Marine Corps experiment [14].
- The simulation must be runnable on low-cost, open-system, multi-platform environments. While the methodology proposed in this paper concentrates on military applications, an operationally-focused simulation is also well suited for emergency management, disaster relief, fighting forest fires, etc. Often the local police and fire units tasked with handling these types of emergencies only have low-end hardware.
- The simulation must be capable of running in multiples of wall-clock time. As described later, in certain applications the simulation must run in near real time, and in other applications it must run much faster than real time.
- The simulation must be able to receive and answer queries from external agents. This capability allows external agents to use the operationally-focused simulation to help monitor the current, ongoing operation for deviations from the plan.
- If needed, multiple simulations should be capable of operating together. While there is no immediate need for multiple, cooperating simulations, this simulation should be compliant with known, accepted protocols so that this capability is not precluded if it is needed.
- The simulation should be based on an aggregate-level model. In military operations, the basic rule of thumb is that commanders fight with units two levels below them: brigade commanders fight with companies; battalion commanders fight with platoons; etc. This level of abstraction is sufficient for the users of the simulation; therefore, in a desire to be able to run much faster than real time, the simulation need not be entity-level.

Many of these requirements for an operationally-focused simulation were later confirmed/validated by Blais and Garrabrants [14]. Surdu, Haines, and Pooch described a prototype simulation implementation that meets these requirements. In addition, their use of VMAP-2™ terrain databases[3] addresses the issue of exercise setup time and cost. This methodology does not rely on the simulation developed by Surdu, Haines, and Pooch; any simulation that meets these requirements could support this methodology.

One government-developed simulation that does not currently have all the properties described but which might be appropriately modified to do so is ModSAF[4]. While ModSAF and its proposed follow-on product, OneSAF, are entity-level simulations, their Distributed Interactive Simulation (DIS) and Persistent Object Protocol (POP) protocols could be wrapped in an "agent" to manage the receipt and answer of subscriptions and queries. ModSAF is not inherently cross-platform, but it has been ported to a variety of platforms, and the GUI (which communicates with the simulation via UDP/IP messages) might be rewritten in a language like Tcl/Tk or Java to provide this capability. ModSAF has provided the basis for the OneSAF Testbed, which is exploring technical issues involved with replacing all previous semi-automated force simulations with a single simulation [15].

While there are a number of useful and excellent training simulations, none are directly applicable for use during operations. There are no known systems for using a simulation to monitor current military operations. The purpose of this paper is to propose a methodology for doing so.

## 4. Proposed Methodology

The proposed methodology is summarized in Figure 1. The methodology involves the interactions of a number of packages and tools, including the operationally-focused simulation discussed in the previous section of this paper, intelligent agents, combat attrition models, path-planning algorithms, etc. Each of the various components of the methodology is discussed below.

**OpSim***:* The operationally-focused simulation runs in near real time, tracking the predicted progress of the plan. The progress of this simulation can be monitored from the Web-based GUI. The Operations Monitors (OMs), discussed below, register interest in various entities and events with the simulation, and they query the simulation directly for information.

**Operations Monitors (OMs)***:* The OMs are the heart of this methodology. They perform two important functions. They take information from WorldView and update the state of entities in OpSim, seamlessly re-synchronizing the simulation to the real world. More importantly, they monitor the progress of the simulation and compare it to that of the real operation. When they discover significant deviations between the real world (WorldView) and the simulated world (OpSim), events referred to as Potential Points of Departure

[3] VPF™ and VMap-2™ are registered trademarks of the National Imagery and Mapping Agency.

[4] Modular Semi-Automated Forces (ModSAF) was originally built by Loral Advanced Distributed Simulations. It is under configuration control by Simulation Training and Instrumentation Command (STRICOM), Orlando, Florida.

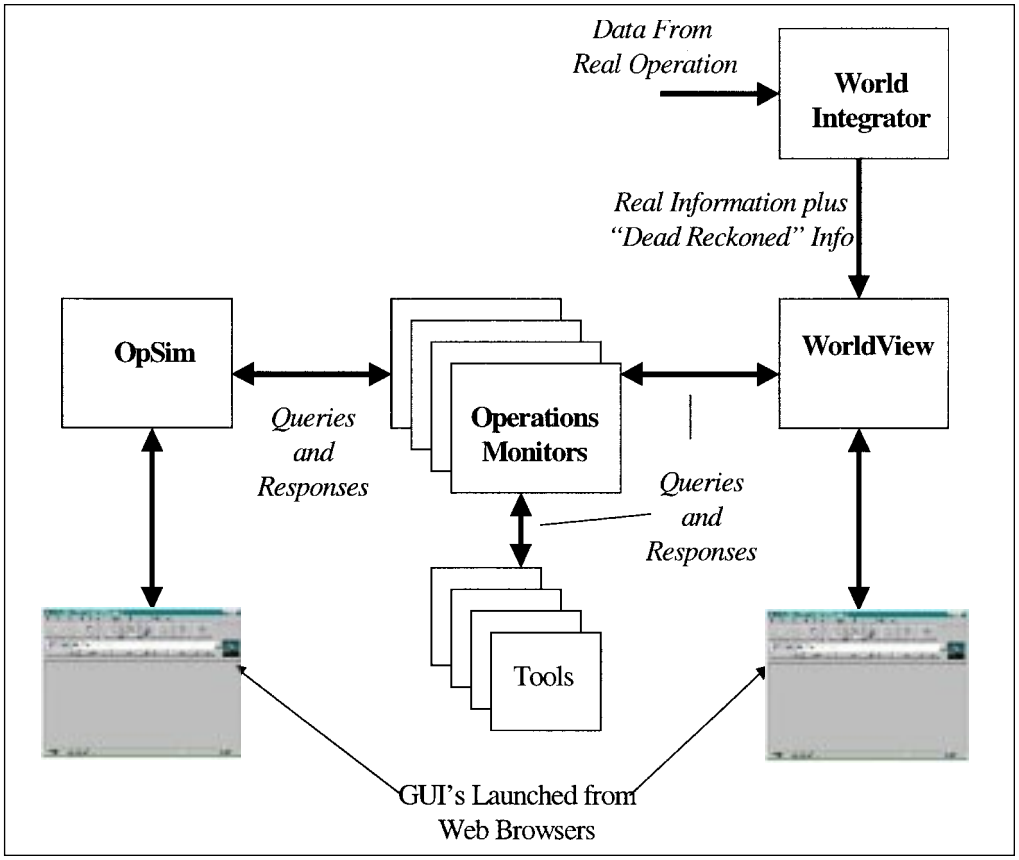


**Figure 1**. Proposed methodology

(PPDs), they launch one of a number of tools to explore the ramifications of these deviations.

It is important to note that OMs do not take actions with regard to the plan; rather, they explore the ramifications of differences between the real operation and the planned operation. The job of OMs is to help human decision makers manage information [16-18]. OMs should be considered part of the team, not a replacement for decision makers [19]. OMs make some judgement about the seriousness of any differences and issue advisories to the decision makers.

Also, it is important to note that OMs must be proactive. It is not sufficient, for example, for an OM to inform decision makers that some planning timetable has been broken. The OMs must look ahead and inform decision makers in advance if some goal is unlikely to be met. For instance, if some future event requires that three of five preconditions be met, the OM must determine whether these preconditions are likely and assess the probability that the eventual goal can be accomplished. When this probability becomes "low enough" the OM must inform the decision maker.

OMs are implemented as rational, utility-based agents. There are a number of useful definitions of what is an agent. Franklin and Graesser [20] proposed a list of characteristics that characterize a piece of software as an agent: autonomous, reactive, goal-oriented, persistent, communicative, adaptive, mobile, and flexible. According to Franklin and Graesser, software which has the first four characteristics is an agent, and software which exhibits some of the other attributes in the list fall into more specialized categories of agents. This method is used in defining OMs within this methodology as agents.

Each OM is interested in only a narrow domain. By focusing each OM on a very narrow domain, the problem of building intelligence into these agents becomes more tractable. The prototype OMs use a simple expert system to reason about their domain and the current operational situation. Promising approaches for improving these agents are hybrid machine learning techniques, such as KBANN or EITHER [21, 22]. KBANN (Knowledge Based Artificial Neural Network) uses a rule-based expert system (something that users may be able to articulate in at least a general way [23]) to initialize an Artificial Neural Network (ANN). After training the ANN, it is sometimes possible to convert the ANN back into an expert system for user verification. This often results in discovering new relationships that were previously unknown to the user, a process known as data mining [21, 24-27]. EITHER is a system that combines inductive and deductive machine learning algorithms to learn general, higher-level (complex) rules from a series of facts and lower-level (simple) rules, stated as Horn clauses. Like KBANN, EITHER has been used to successfully improve the accuracy of a rule-based expert system [22].

PPDs are not always held in fixed, global knowledge bases. Instead they are domain-specific. Each OM has a knowledge base that it uses to analyze the discrepancies between the real operation and the simulated plan. Because of the inherent uncertainty in the knowledge associated with the domains, non-crisp forms of reasoning (or soft computing) often are required. For some OMs, the PPDs may reside in a fixed knowledge base, which may be in the form of rules [23], while the PPDs for others may be in the form of some refinable domain theory [21, 28], or any other representation scheme appropriate for the domain. It is not the intent of this methodology to impose any particular inference mechanism on the OMs. The prototype OMs constructed to validate this methodology used fuzzy rule bases [29]. With a fuzzy rule base, it is relatively easy to capture the knowledge of domain experts and later explain how the OM made the decisions it made. *The point is that the specific inference mechanism used to implement a particular OM is independent of the overall methodology proposed in this paper.*

**WorldView**: The WorldView module is a representation of the real operation. In order to make the job of the OMs easier, the representation of the state of the real operation and the simulated plan should be as similar as possible. WorldView receives information about the state of the real operation through a series of APIs. It then transforms this information into a form that the OMs can easily interpret.

***WorldIntegrator***: WorldIntegrator has the onerous task of monitoring the real operation, processing that information, and passing it to WorldView. In some systems, such as the Global Command and Control System (GCCS), this may involve querying a database. In other systems, this may require "eavesdropping" on the network. The reason for this intermediate step is that in real operations, reports on some entities may be intermittent. It is the job of WorldIntegrator to "dead-reckon" these intermittent reports and pass them into WorldView. Clearly, when an entity has been "dead-reckoned," this must be reflected in the information that WorldView gives to the OMs.

The issue of WorldIntegrator and WorldView involves sensor, data, and information fusion. WorldIntegrator must determine when an entity has been unconfirmed long enough that its actions must be dead-reckoned. When some sensor reports a similar unit, WorldIntegrator must determine whether this is merely the lost unit reappearing or a different unit. These and other issues regarding sensor, data, and information fusion are open research issues. Since no such system is currently available, for purposes of this research a combat model simulates the functions of WorldView and WorldIntegrator in the prototype.

**Tools**: This paper does not attempt to enumerate all possible, useful tools. Instead, it gives examples of tools and how the OMs might use them. For example, the Enemy OM might note, based on information from WorldView, that there are two enemy mechanized battalions in the area of operations rather than the one assumed during COA analysis. The OM can call a combat attrition model to determine the difference in expected losses, or the OM might merely apply a closed-form solution to Lanchester equations to get a quick estimate of expected losses. If it appears that this difference will adversely affect the plan, the Enemy OM will notify the decision maker.

Similarly, if the Mission and Time OMs note that some ground unit had missed an important phase line by 45 minutes, they might launch another simulation to explore how this would affect other units. If the

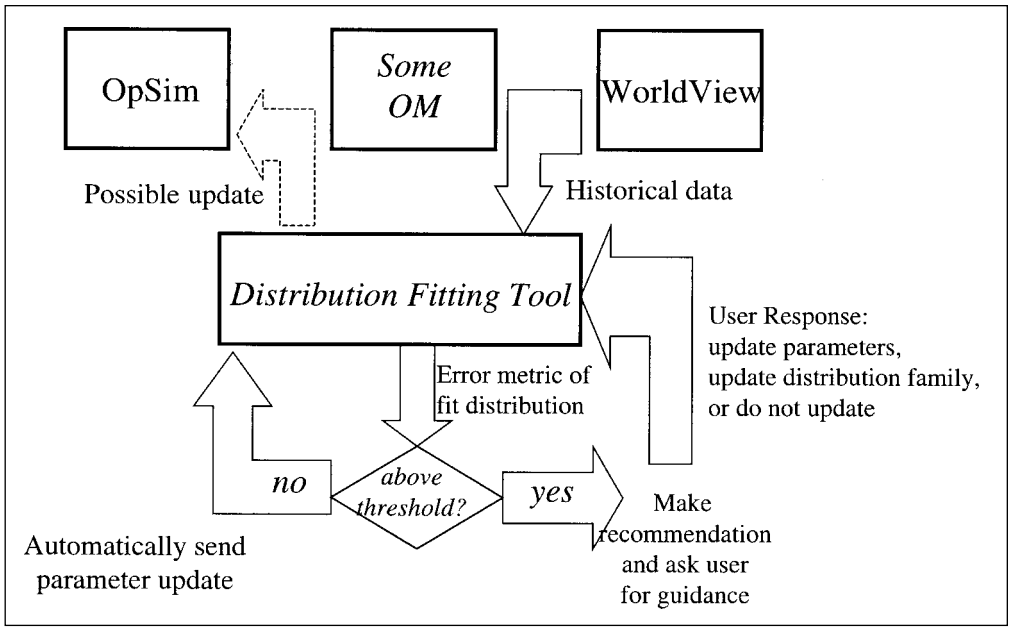


**Figure 2**. General depiction of synchronization/updating scheme

effect is minimal, the OM might recommend to the commander that the overall time line be shifted 45 minutes to resynchronize the simulation.

There is a great deal of interaction between the OMs and OpSim and between the OMs and WorldView. This is conducted through a message-passing protocol. There are two kinds of requests: individual queries and registration of interest (subscriptions). An OM, for example, might send a query to WorldView and OpSim about the status of a particular unit. This is done as an individual query. An OM might also register interest in certain information. For instance, the Troops OM might register for periodic updates of the strengths of units. Registration of interest is preferred, since on an ongoing basis, it requires roughly half the number of messages as individual queries. OpSim launches a separate thread to handle each of these registrations.

Note that normally OMs do not make tactical decisions. The purpose of an OM is to explore differences and report findings. The autonomy of the OM lies in its ability to decide when and if to launch other tools. As noted in the DARPA CPoF concept, battlefield visualization tools should be decision-centered. Among other things, this means that these visualization systems "show decision-relevant details, highlight relevant changes, anomalies, [and] exceptions, and portray uncertainties" [3]. These are exactly the pieces of information that our proposed methodology is designed to provide. Visualization is not a tool to show the battlefield in a unique way; visualization is a process that occurs within the heads of the commander and his staff [30]. Our proposed methodology provides additional support for this process.

## 5. Synchronizing the Real Operation with the Simulated Operation

In order for the OMs to adequately compare the real operation with the simulated operation, the two representations must be "close." An axiom in military planning is that no plan survives the first rifle shot. Once the operation commences, the plan will certainly diverge to some extent from the real operation. The job of the collection of OMs is to identify when this divergence has become so great that the success of the operation is in jeopardy. The OMs report this concern to the decision maker (probably with some rating of certainty attached to this conclusion).

Once the decision maker has been notified that the currently running simulation no longer accurately reflects the state of the actual operation, the simulation should be updated. If the simulated plan is allowed to continue to diverge from the real operation over time they will become almost completely unrelated. Any analysis the OMs would perform at that point would be meaningless. This updating also allows OpSim to better predict where the operation will be at some future time. The problem, however, is to define a synchronization mechanism which is feasible and adaptive. The proposed approach is best illustrated through two examples.

The combat effectiveness of entities (units) in the simulation is characterized by some probability distribution(s). These probability distributions (which might be different for different classes of entities) are used in the analysis of the various COAs during the planning phase, and they are also used to simulate the interactions between the various entities as the simulation was paralleling the operation in near real time.

At the time the OMs determine that the real operation and the planned operation are significantly different, they have a body of historical data on the actual effectiveness of the classes of entities. The OMs must do two things: update the current state (e.g., the number of casualties) and update the future performance of the entities within the simulation. An OM tries to refit the historical data to the family of probability distribution described for that class of entity. There are a number of distribution-fitting packages on the market which accomplish such a task. They usually use maximum likelihood estimators (MLEs) to recommend some distribution for a set of data based on which distribution has the smallest p-value or sum of squared errors. When the OM tries to refit the historical data gained thus far in the real operation to the family of distribution defined for the combat effectiveness of the entity, some p-value or sum-of-square errors will be generated.

If the measure of error is below some threshold, the OM will conclude that the distribution family is correct, but that the parameters (e.g., $\mu$ and $\sigma$ for a normal distribution) are incorrect. In this case, the OM would automatically send an update message to OpSim. As this should make the future performance of the simulation better over time, this updating scheme makes the system adaptive. If the error metric is above that threshold, the OM will conclude that the family of distribution chosen is incorrect. At this point, the OM will open a dialog with the user of the system describing the problem. It is then up to the user to decide whether to merely update the parameters of the current distribution with the best possible values, change the family of distribution used, or determine that the results are anomalous and decide that no update is necessary.

As an example, in OpSim combat effectiveness is represented as a distribution family (e.g., Normal or Exponential) with some parameters. Periodically (at a time interval determined by the user, but approximately every five minutes by default) the Results OM queries the simulation and the real world for the historical data on the various classes of entities (e.g., Bradley mechanized infantry platoons). Using the real-world data as the "true" distribution, the Results OM conducts a Kolmogorov-Smirnov goodness-of-fit test of the data from the simulation [31, 32]. (The more common Chi-Squared goodness of fit test has also been implemented, but the Kolmogorov-Smirnov test has

greater power, in the statistical use of the term.) When the data from the simulation fit the real data with an alpha value of 0.95, the Results OM takes no further actions. When the data from the simulation do not fit the real data, the Maximum Likelihood Estimator (MLE) of a variety of distributions is computed for the simulation data. Then sample data streams are generated with each of the computed MLEs, and a Kolmogorov-Smirnov goodness-of-fit test is computed against the real data. The new distribution that has the best fit is nominated as the new distribution. If the new distribution is in the same distribution family as the true data, the Results OM tells OpSim to update the parameters of the distribution. In order for the system to remain stable the new value is set to the average of the old value and the suggested value. This allows the parameters to move slowly toward the "true" value, rather than changing rapidly and giving undue importance to transient spikes in the data. In experiments in which the probability distribution is intentionally set to a different family of distribution, the Results OM tells the user what the old family and parameters are as well as the new recommendation, and asks the user to confirm or cancel the update. Over the course of an experiment, the parameters that describe entities in the simulation gradually converge on those of the real operation. Most of the changes occur immediately after small engagements, after which new data is available to the Results OM.

The preceding illustration shows adaptive updating of OpSim for combat effectiveness; however, the methodology is valid for other characteristics as well. Movement rates of classes of entities are also defined in terms of probability distributions (e.g., an average movement rate and some variance). If all the entities (units) of a certain type are moving more slowly than predicted, we would want to do two things: put the entities in their current locations in the real operation and adjust the parameters on the distribution which describes their movement rate. Again, it would be up to the decision maker to determine if this difference was anomalous (e.g., unseasonable weather, entities beginning movement late, etc.) or required an update to the simulation.

Another, less technically interesting update of OpSim would occur when the number of entities was significantly different. If, for instance, the plan assumed that the enemy would have three tank battalions, but WorldView indicates the enemy actually has four, OpSim would need to include this additional unit in the future. Adding this unit automatically would be difficult, since an intelligence officer would have to provide OpSim with an estimated plan for this new entity. Initially, the OM might provide OpSim with a plan that extrapolates the entity's velocity vector, but adding new entities would probably need some human intervention.

Prototype Friendly and Enemy Forces OMs have been implemented. These are identical, except that one looks at the friendly forces and one looks at enemy forces. Each of the Forces OMs periodically compare the number and strength of units (friendly or enemy as the case may be) in the plan (simulation) versus those in the real operation. Almost as soon as the operation begins, the strengths of the units in the simulation and real operations begin to diverge. In test scenarios additional enemy units have been added to the real world to force the Enemy Forces OM to analyze the impact. The Forces OMs use a fuzzy rule base to determine when the strength and/or number of units are significantly different. When the strength and/or number of units are significantly different, the Forces OM must determine whether this difference is important. It does this by feeding the current, real situation into a simulation (another instantiation of OpSim) and running that simulation to completion some number of times (determined by the user). The mean results of the new simulation are compared with those of the planned operation. Again this comparison is done using a fuzzy rule base. If the Forces OM determines that the impact of the change in strength/number of the units involved significantly impacts on the probability of mission accomplishment (or the end strengths of the friendly forces after the operation), the user is notified. Note that the prototype uses two small, fuzzy rule bases to conduct its analysis: one to determine if there is a significant difference between the real world and the simulation and one to determine if the change in forces has a significant impact on the outcome of the operation. If the difference is insignificant and only involves the strengths of units (e.g., the unit is at 50% strength rather than 75% strength), the Forces Monitor sends an update message to OpSim. If the difference is significant or involves adding or deleting units, a human must make the corrections manually.

The basic idea, therefore, is for one or more OMs to analyze the performance of the simulation with respect to the real operation. The OMs can make small updates in the parameters of OpSim automatically. For larger deviations they query the user for help. The amount of data that needs to be gathered before the differences are significant is unclear. One problem with analysis of military operations is that the experiments are not reproducible, much of the area of operations is destroyed in the process, and many of the witnesses are killed. One approach to dealing with this issue is for the threshold (used to determine whether to update automatically) to be adaptive. The OMs can use the performance of the simulation after an update to help it adjust the threshold. At some point it might also be appropriate for the correctness of the human user's decision to be used to adjust this threshold as well.

## 6. How OMs Are Created Dynamically

As stated earlier, OMs focus on a narrow domain. This makes their design and implementation more

**Table 1**. Sample structure of tool classification used by an OM

| | |
|---|---|
| Domain 1: | |
| Tool 1 | Memory(N), O(N), Bandwidth(N) |
| Tool 2 | Memory(N), O(N), Bandwidth(N) |
| Tool 3 | Memory(N), O(N), Bandwidth(N) |
| Domain 2: | |
| Tool 4 | Memory(N), O(N), Bandwidth(N) |
| Tool 5 | Memory(N), O(N), Bandwidth(N) |

tractable. When the system is first launched, a manager OM creates the first layer of OMs in the hierarchy. The overall manager is responsible for synthesizing the reports of the agents below it in the hierarchy. The first layer of OMs in the hierarchy compares the current situation with the plan, each looking at the operation from a particular, narrow perspective. One such taxonomy for OMs in this first layer is the use of the Combat Functions (as defined in FM 100-5): maneuver; fire support; air defense; command and control; intelligence; mobility, counter-mobility, and survivability; and combat service support (logistics and personnel) [33].

These OMs in the first level of the hierarchy have a number of tools (and additional agents) available to them to perform their analysis. Each OM uses a rule-based expert system to make inferences, determine whether to launch additional tools to help with analysis, and decide what actions to take. There are basically three types of rules: those that dispatch other agents or tools, those that report information to other agents, and those that take other actions (e.g., advising the human of problems or updating the simulation as discussed earlier). Some of these rules are domain-dependent; they are related to the particular focus of the OM. Other rules are domain-independent; they are related to general issues, such as the resources needed by a particular candidate tool. Domain-independent rules take into account RAM usage, time complexity, bandwidth, etc. (as shown in Table 1). These domain-independent rules are important, because they help ensure that the system does not grind to a halt during times of peak activity during the operations. For instance, the eventual system might have two different path planning algorithms that could be used to determine the impact of rerouting logistics. If one algorithm has a smaller time complexity but is somewhat less accurate than the other algorithm, the OM might choose to use this algorithm if the host on which it is running is already very busy. As mentioned earlier, OMs are utility-based agents, and these domain-independent rules provide information used to compute the utility of a particular action.

One possible taxonomy for agents in a second layer of OMs might be along the lines of the Army's METTTC mnemonic (Mission, Enemy, Time, Troops, Terrain and Weather, Civilian Impact). Under this taxonomy, one OM would be looking for differences in the size, strength, and/or composition of the enemy. Another might be looking at effects of terrain and weather.

One possible, partial expansion of an OM hierarchy is shown in Figure 3. Note that an instantiation of a particular class of OM can exist at multiple points in the hierarchy. The only restriction is that an OM cannot call a tool above it in the hierarchy to avoid circular dependencies. Since any OM can create any number of subordinate OMs to aid in its analysis based on the current situation and delete OMs that are no longer necessary, the hierarchy is dynamic.

This discussion of OMs has repeatedly referred to the use of rules. This is for ease of discussion. This methodology does not require nor rely on any particular reasoning mechanism. As discussed earlier, OMs

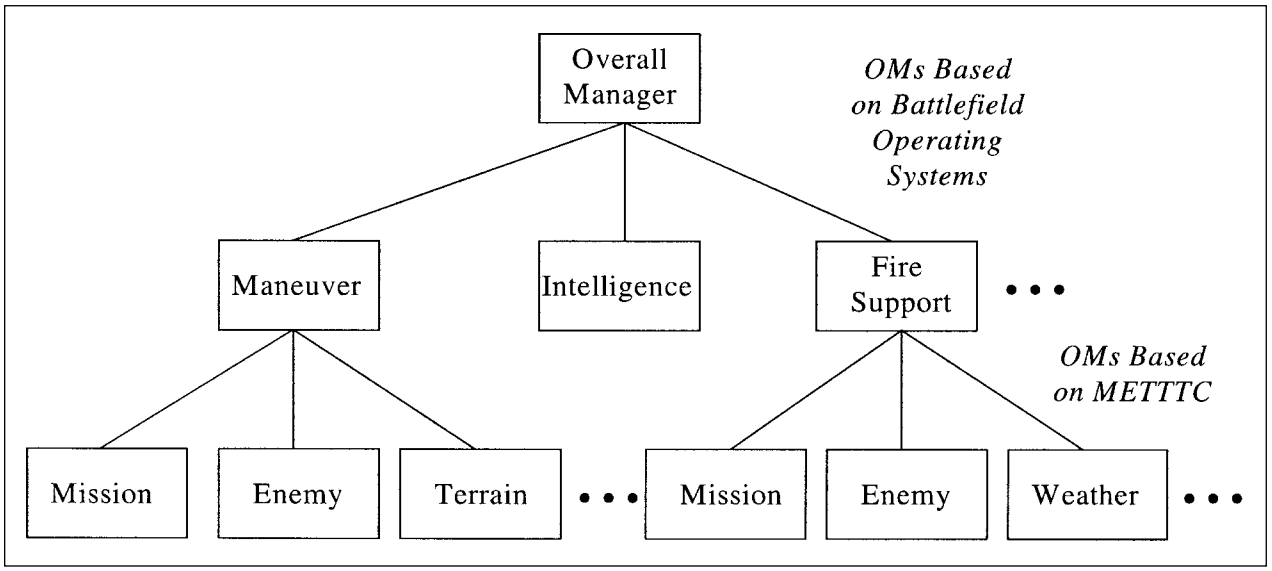


**Figure 3**. A possible, partially expanded hierarchy of OMs

**Table 2**. Sample monitor configuration file

| Launch | Java Class Name | Mission | Battlefield Function | Complexity |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| AUTO | OpSim.ManueverMonitor | * | maneuver | COMPLEXITY MEMORY low TIME medium |
| AUTO | OpSim.IntelligenceMonitor | * | intelligence | COMPLEXITY MEMORY medium TIME high |
| AUTO | OpSim.FireSupport | defendInDepth\|defendInSector | firesupport | COMPLEXITY MEMORY low TIME low |
| MANUAL | OpSim.Engineer | * | m/cm/s | COMPLEXITY MEMORY low TIME high |

could use a variety of reasoning technologies, including crisp rule bases, fuzzy rule bases, machine learning techniques, artificial neural networks, etc. The OM's domain and mission will probably dictate the appropriate reasoning mechanism. OMs must be capable of launching other tools as necessary, making inferences about the current situation, and taking actions as appropriate. This methodology does not specify the technology used to perform these tasks.

To avoid the haphazard and ad hoc creation of OMs, a general principle was devised. This is based on something that every commander readily has available: his mission. Army manuals clearly define the exact meaning of various tactical tasks, such as seize, secure, defend in depth, and screen [33]. For instance, while it is not stated in this bullet format, a seize mission involves:

- Gain control of a piece of terrain
- Deploy to prevent its destruction or loss to enemy action.

and defend in depth involves:

- Prevent enemy forces from penetrating the rear boundary of the sector.
- Retain flank security.
- Maintain integrity of effort with parent unit's scheme of maneuver.

While there is not a one-to-one mapping between mission statements and what OMs create automatically, the mission provides some guidance for the top-level OM. For instance in the prototype implementation, the seize mission was used in an experiment. Some analysis of a generic seize mission by domain experts (field grade Army officers at Texas A&M University) indicated that the Maneuver OM, C2 (Command and Control) OM, and Intelligence OM would be needed for all seize missions. The OMs representing the other Battlefield Functions (Air Defense; Logistics; Fire Support; and Mobility, Counter-mobility, and Survivability) would be needed on a case-by-case basis. A similar analysis was done of defense in sector, and Fire Support OM was added for that mission. Below the top-level OM, the other OMs create sub-OMs as their reasoning mechanism dictates.

In the prototype implementation, each type of OM has a configuration file that is read when the OM is created. An example of a portion of the configuration file for the top-level OM is shown in Table 2. The item in column one indicates the subordinate OM that should be automatically launched if the mission of the operation matches one of those listed in column three. The second column indicates the Java class name used for dynamic instantiation of classes. This allows a new monitor to be added to the system by adding its class name to the configuration file without modifying or recompiling the code. The third column indicates missions for which this monitor should be automatically instantiated. If the keyword MANUAL was listed in column one, column three is ignored. The fourth column is for use by the specific OM, and can be used differently by different classes of monitors. Since this segment of the configuration file was for the top-level OM, this lists which Battlefield Function this OM addresses. This is analogous to SYMPTOM in Table 1. This column is important, because there may be two or more tools that address the same Battlefield Function, but have different space or time complexities. At this point in the prototyping stage, all OMs are assumed to be equally accurate, but a rating of the tool's proficiency will be included in future prototypes. According to the proposed OM taxonomy shown in Figure 3, for OMs in the second layer, column four would contain elements of METTTC. Finally the items in column five describe the space and time complexities of the OM. This is used for the domain-independent reasoning discussed earlier. Since fuzzy rule bases are used in the prototype, the semantic labels low, medium and high are used.

In the prototype implementation, once the default (or automatic) OMs have been created, the user is presented with a GUI. From this GUI the user (commander) can manually launch other OMs as desired. For instance, if the enemy had a credible air force, the commander might want the Air Defense OM running, and he could do this from the GUI. This allows the OMs to be automatically launched according to the principle described earlier (based on the mission) but allow flexibility for the commander to tailor the system to the particular operation.

## 7. Implementation

### *7.1 Client-Server Architecture*

The OpSim server listens on a known TCP port for connections. When a connection is requested, a new simulation object is created as a thread (lightweight process). Multiple copies of simulation objects can be running simultaneously. When a simulation object is created, it is given two port numbers. The simulation will establish server Transport Control Protocol/Internet Protocol (TCP/IP) sockets on these two ports. The first is the port that will be used by the GUI to connect to the simulation. This connection is the control connection and is used by the GUI to give commands to the simulation, such as start, stop, pause, and resume. The GUI and all other clients use the second connection to subscribe to information. With each connection request to this port, which is a subscription request, the simulation creates a new subscription listener thread. Each client that subscribes to information from the simulation has its own connection and its own servicing thread on the server side of the simulation. The GUI, therefore, has two connections to the server—one on which to send control commands to the simulation and one on which to subscribe to information and send one-time queries.

While the simulation itself runs on a server machine (which may have significant computation power if required), the interface to the simulation is a Java Applet which can run on relatively low-end machines anywhere on the Local Area Network (LAN) and viewed in any Web browser with a Java virtual machine. As part of the Java security management scheme, Java Applets can only connect to machines from which the applet code was downloaded. For this reason, a small HTTP server is required to run on the same machine as the simulation server.

The connection architecture is shown in Figure 4. The simulation sees Operations Monitors as just another subscription connection. While these connections are routinely called "subscription connections," it should be noted that these same connections are used for the client to transmit one-time queries and for the server to reply to those queries.

This architecture was chosen so that the clients could be very thin. The GUI client only conveys commands from the various buttons and boxes on the GUI, through API calls, to the simulation, and it displays information that it has subscribed to on the screen. The movement of entities, the resolution of interactions (e.g., combat), and the management of time are resolved by the simulation on the server machine.

Control over the simulation is exercised by the user, through the GUI, which communicates to the simulation through an API. For instance, a scenario is represented in OpSim as a plan file. This plan file is critical, because it contains the list of entities and their planned actions. The plan file resides on the local (i.e., client) machine, is read by the GUI, and is transmitted to the server. This was done so that each client (human) could control the management of scenario files locally. Maintaining the plan files locally requires the GUI applet to read the file from the local file system. Normally Java Applets are not allowed to access the local

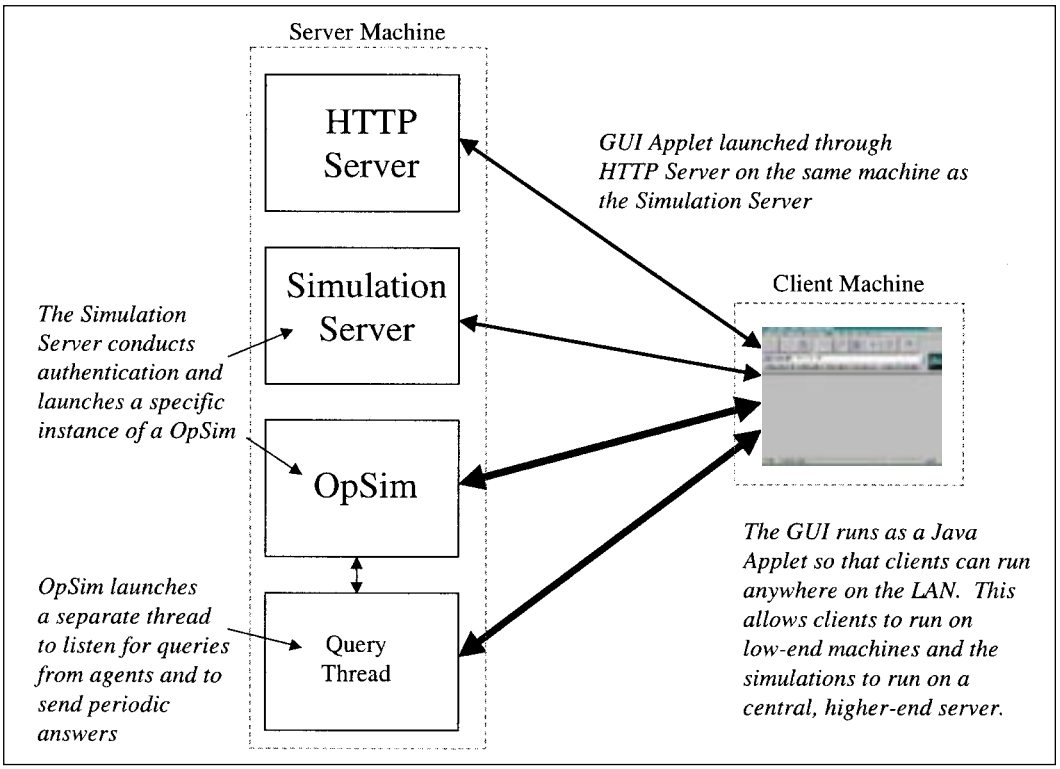


**Figure 4**. Connection architecture for OpSim

file system. In order for the GUI applet to read the plan file, the GUI applet is a signed Java Applet. While the plan file resides only on the client machine, the terrain database physically resides both at the server and client machines. This is due to the large size of the database and its static nature.

Information is passed between the client(s) and the server via a simple message passing protocol so that neither the server nor the clients is restricted to any particular language or object broker architecture. If a developer wanted to build a client in a language other than Java, the developer would only need to connect via a standard TCP/IP connection to the server and implement the message passing protocol to do so.

### *7.2 Terrain Representation*

Most military computer simulations today read in terrain data, translate it into a proprietary format, and then use the proprietary formatted data. This occurs due to different aspects of geographic information being in different formats, such as elevation posting being in Digital Terrain Elevation Data (DTED) format and natural and cultural features being in Digital Feature Analysis Data (DFAD) format. Since OpSim was designed for the operational environment, an involved process of converting various data sources into a unified database is impractical. For the sake of performance and usability, a better representation of terrain data for use in OpSim was needed. The National Imagery and Mapping Agency (NIMA) (formerly the Defense Mapping Agency) has a viable solution [34].

OpSim uses terrain data in Vector Product Format (VPF™)[5]. VPF is a standard format, structure, and organization for large geographic databases and is based on a relational data model. VPF allows applications to read terrain directly from computer-readable media (e.g., CD, disk, or tape) without converting it to an intermediate format. Terrain data in VPF is in vector format that concerns itself with vertices, lines, and three-dimensional coordinates. This allows for data to be accessed by spatial location and thematic content (an advantage over the raster model for terrain data) [35, 36].

The specific VPF product used in OpSim is VMap-2™. VMap-2 is intended for use by tactical planners and provides terrain details at scale 1:50,000 and scale 1:100,000. OpSim uses a prototype VMap-2 database that includes data libraries for Ft. Hood, Texas, and Norfolk, Virginia. Each library contains 12 thematic coverages, such as boundaries, elevation, transportation, and vegetation [37]. The design of OpSim envisioned the need for a file containing changes in the database [13]. This delta file would be used to record destroyed bridges, new power lines, flooded areas, etc. This would facilitate the dynamic updating of a mostly static database. This feature has not been implemented in the prototype OpSim.

It is important to note at this point that there is no consensus on terrain format. VPF is an emerging standard, and there are a number of competitors. The Defense Modeling and Simulation Office has a terrain format, called SEDRIS. The purpose of SEDRIS was to have a uniform terrain format that people could convert their proprietary terrain to and from. If an organization wanted to conduct an exercise using both ModSAF and BBS, and the correct terrain database only existed for ModSAF, for instance, rather than paying for someone to build the BBS terrain database, the ModSAF database would be converted. The ModSAF-to-SEDRIS converter would be used, and then the SEDRIS-to-BBS converter would be used. In this way, each simulation would need to build two converters (to and from SEDRIS) rather than build a converter for every simulation with which it might likely need to interact [38]. There is some wisdom in OpSim eventually being able to read SEDRIS data as well as VPF. Why not take advantage of terrain databases that someone has already spent money to build? NIMA claims that eventually all their digital mapping products (such as DTED and DFAD) will be converted to VPF. As VPF becomes more widely accepted and terrain databases exist for many more areas, this seems to be the most practical solution for the tactical, operational environment.

### *7.3 Near Real Time Constraints*

OpSim should be able to run as fast as possible when analyzing courses of action and in near real time when the actual operation is in progress. To this end, there are four clock modes in OpSim, and the user can change the clock mode during execution. The first clock mode, "fast," allows the simulation to advance to the next event on the event queue as soon as it has finished processing an event. This is the mode used by Operations Monitors to explore the ramifications of differences between the planned operation and the current operation. It is also the mode used to analyze a variety of courses of action during planning.

When watching the simulation running in fast mode on the GUI, one sees snapshots (at the update rate set by the GUI) of whichever course of action is being executed at the time. One might see an experiment near the end of execution in one screen update and the next experiment near the beginning of its execution in the next update. This view is often counterintuitive to a viewer, but it generates information about the courses of action very quickly—which was its intent. The second clock mode, "presentation mode," was created to allow a viewer to step through the simulation very quickly, but slowly enough that actions happen in sequence. In fast mode, a friendly brigade attacking an enemy battalion usually takes less than two seconds to simulate a two-hour operation. In presentation

[5] VPF and VMap-2 are registered trademarks of the National Imagery and Mapping Agency.

mode, it takes about ten minutes to view this same two-hour operation.

The last two modes are near-real-time modes. They are designed so that the simulation will be executing the plan at the same rate that the real operation is progressing. In this mode, an event will not be executed prior to its real-world time, although it may be executed slightly later. If the server machine is very busy there is the potential in this mode for the simulation to fall behind the real operation by a significant amount. This might make the Operations Monitors detect differences between the plan and the real operation that do not really exist. For this reason, the user has the option of two near-real-time modes. The first mode, "keep," indicates that the user is willing to fall behind temporarily on the assumption that when the peak load on the machine has passed it will catch back up. The second mode, "discard," indicates that the user is willing to discard some events in the interest of remaining close to real time. Both modes have obvious drawbacks, but rather than make the choice between them as a design decision, OpSim leaves this decision to the user.

### *7.4 Attrition Modeling*

It was not the purpose of this research to propose a new and better attrition model. A goal of the OpSim design was to make OpSim flexible enough that it did not depend on any particular attrition model. To this end, an abstract class, AttritionModel, was created, and two specific attrition models which inherited from AttritionModel were built. The first was the Dupuy QJM model [39, 40]. The second was an ad hoc model based on recreational war games. The QJM model is a deterministic model, and the basic model was modified slightly to make it stochastic. In addition, the QJM model was designed for large forces (e.g., divisions and corps), so some modifications needed to be made in order to apply it to platoons.

The AttritionModel class specifies methods that compute the casualty rates of the two forces. Given a specified duration of the fight, a series of other methods computes the actual number of men, armored personnel carriers, tanks, wheeled vehicles, air defense weapons, anti-tank weapons, and infantry heavy weapons that are destroyed within that time interval. These losses are initially stored in temporary variables in each entity. Once all combats have occurred for that time cycle (all attack events with the same time stamp) are executed, all entities that participated in attacks update their permanent variables. All events with the same time stamp are assumed to happen simultaneously in a discrete event simulation. This process ensures that all entities have the same strength at the beginning of all attack events with the same time stamp.

In OpSim, entities have a rating of their effectiveness, called the Combat Equivalence Value or CEV. OpSim interprets this CEV as the mean performance of the entity. The probability distribution associated with CEV is stored in the entity as well. In the current implementation, CEV can be a parameter of either a Normal or an Exponential distribution. For distributions that require a second argument, each entity also has a CEVSigma parameter. When entities participate in attack events, random numbers are generated according to the specified distribution and parameter(s). These numbers are used to modify the combat results. For instance, an M-1 tank platoon has a Normal distribution with CEV of 1.2 and a CEVSigma of 0.2. In the two implemented attrition models (the ad hoc and the QJM) a random variate is generated according to this distribution, and that number is multiplied by the strength of the unit to determine its final strength.

### *7.5 Performance*

The OpSim prototype has capabilities that focus on the planning and execution phases of an operation. As the planners develop several friendly and enemy courses of action, these can be created in OpSim as described previously. The user can then choose which (some or all) of the courses of action for each side to simulate. The user also chooses how many iterations of each plan to run. On a Sun Ultra 5 workstation, each iteration of a brigade attack on an enemy battalion takes approximately two seconds. So with two sides, having four plans each, for ten iterations, it takes OpSim approximately 320 seconds—just over five minutes. Once the simulation has run all these experiments, a window is created which summarizes the statistics for all the selected iterations. This is shown in Figure 5.

Each entity has some number of personnel, armored vehicles (tanks), armored personnel carriers, unarmored vehicles, air defense weapons, anti-tank weapons, and infantry heavy weapons (e.g., machine guns, small mortars, etc.). During the execution of an experiment, entities report losses to a statistics collector object. After all the experiments are done, the statistics collector computes means and confidence intervals around those means of the losses each side took. For instance in Figure 5, Side A, "Good Guys" lost an average of 213.0 +/– 42.91 personnel when running plan 1 versus Side B's plan 1. The confidence interval is important, because it gives the user some indication of the variability of the results of the individual experiments. Each of the cells in the display is implemented as a list box, so if the user clicks on the cell, the losses from each experiment are shown. See Figure 6.

Along the right column of the statistics window are the summaries across all enemy plans. These are the means of the means, so to speak. This gives some indication to the user how well a friendly course of action performs across all likely enemy courses of action. Again, by selecting on a cell, the data used to compute the cell value are shown.

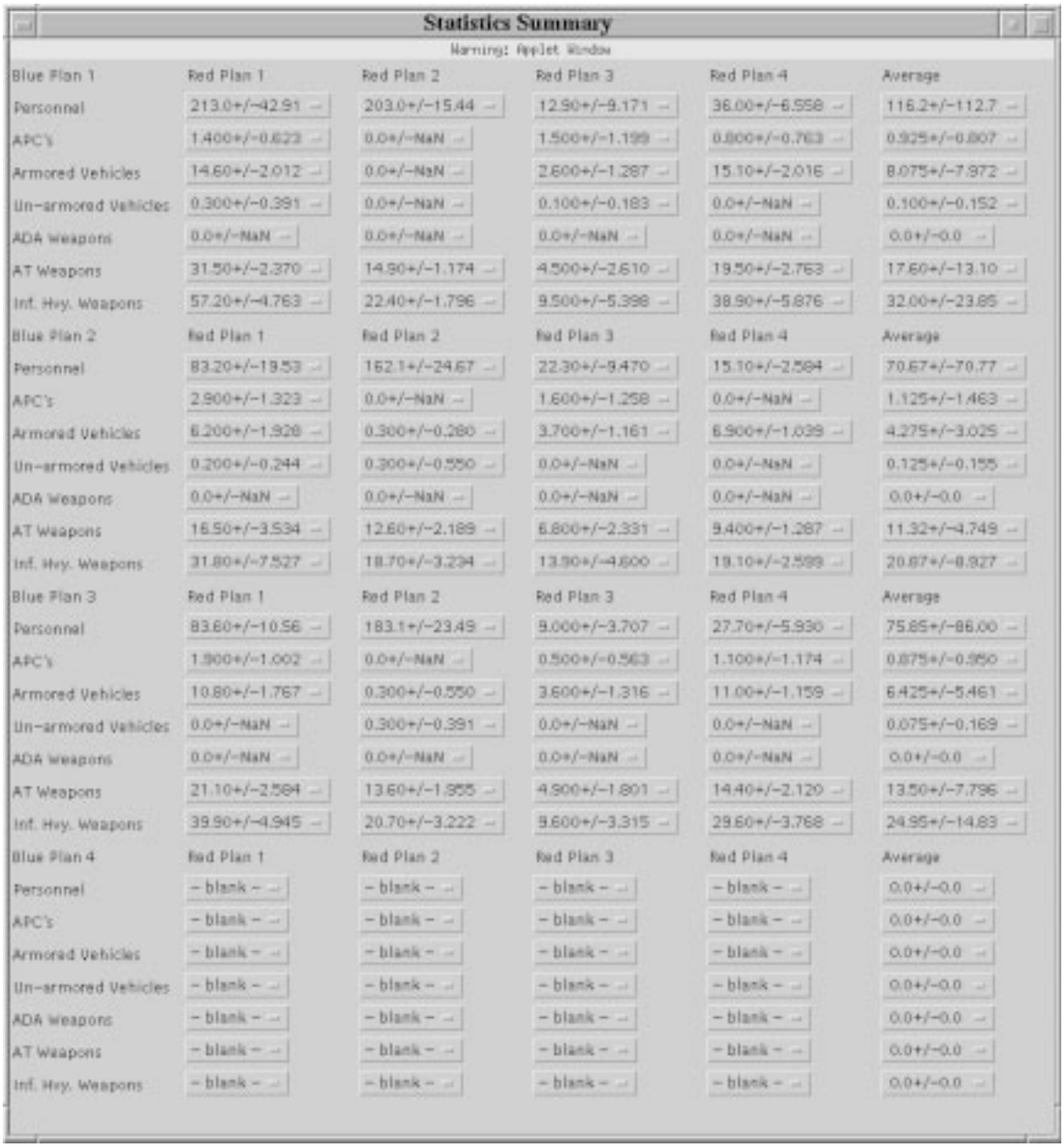


**Figure 5**. Summary statistics created by OpSim

The intent of this statistics window was to provide some data that could be used along with other decision criteria in selecting a course of action for the upcoming operation. Obviously casualties are not the only factor in choosing a plan. There are several lists of factors that are often used to evaluate plans. Among these are the Principles of War, Tenets of Army Operations, and the elements of combat power [33]. Staffs generally use tools, called decision support matrices [41], to help evaluate alternative courses of action, and the statistics generated by OpSim might be used as one of several evaluation criteria.

OpSim is written in Java, which is an interpreted language. Interpreted languages are generally slower than compiled languages; however, since the Java standard has stabilized at Version 1.2, the developers have had time to begin to optimize the Java compilers. In addition, just-in-time compilers are becoming available, many for free, which compile the Java byte codes into machine-specific binaries during execution. In addition, since OpSim was written as a prototype, there was little emphasis on performance optimization. Despite the slow speed of OpSim relative to how fast it *might* be after some compiler and code optimizations, the current implementation is fast enough to allow staffs to quickly evaluate many courses of action and also experiment with some branches and sequels. The analysis done by OpSim is less susceptible to personal bias, group-think, and the other issues discussed in Section 2. In a production system, *the planning capabilities provided by OpSim would greatly enhance the ability of staffs and commanders to make rapid tactical decisions*.

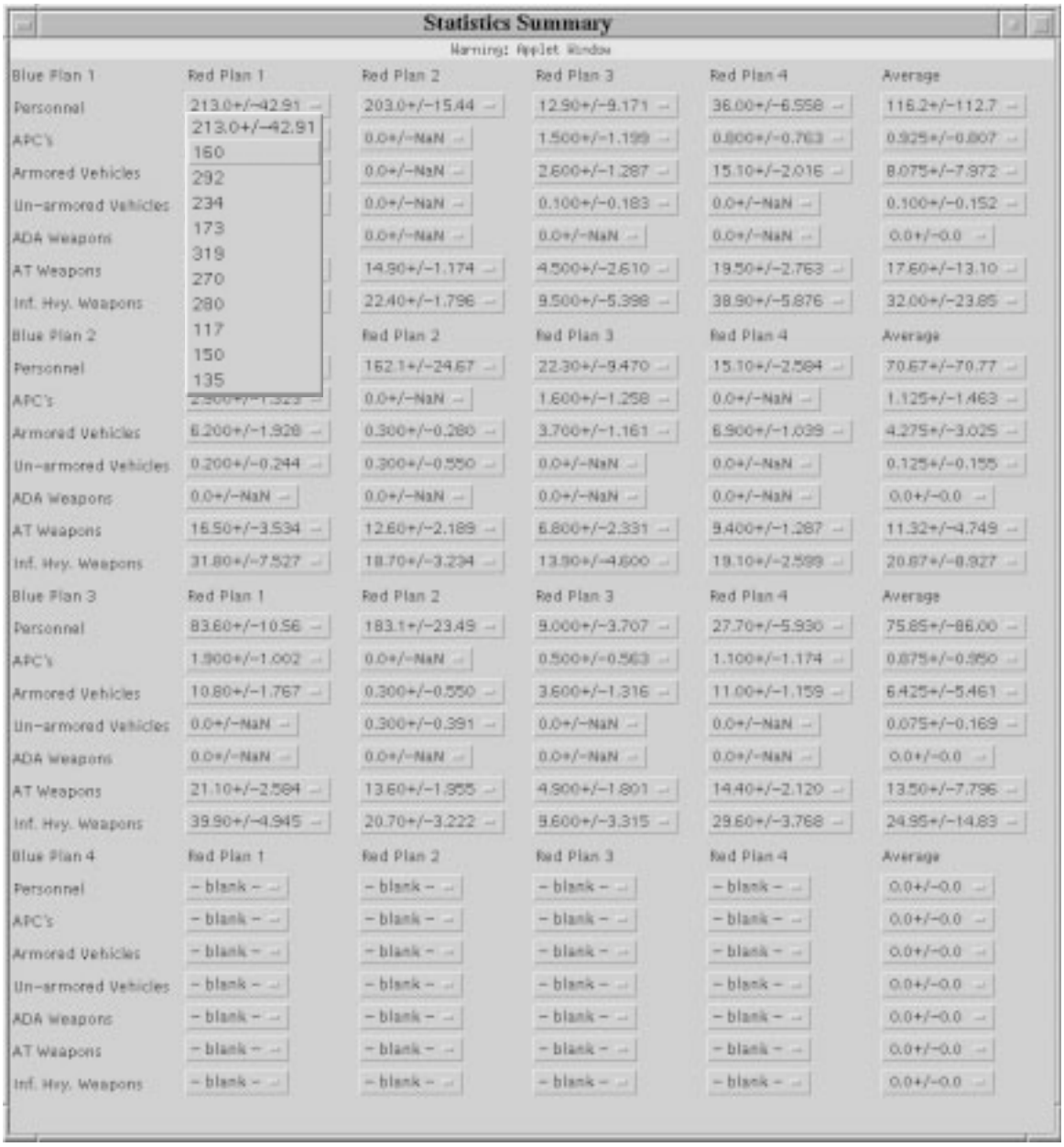


**Figure 6**. Personnel losses expanded to show the results of each experiment

Once the commander has chosen one of the courses of action to execute, OpSim can then be used to monitor the current operation. OpSim incorporates a number of capabilities that facilitate this use. As described earlier, any number of clients can connect to the simulation and subscribe to information. The user can set the clock to one of the real-time modes. In one of the real-time modes, the plan (simulation) should progress at the same rate as the real operation. If the real operation is falling behind or getting ahead of the plan, the clients can get this information from the simulation (and the representation of the real operation). Another useful feature of OpSim is some user-level control over the generation of random variables.

There are two random number generation modes. OpSim wrapped the Java Random class in another class, OurRandom. This was done so that random number generators other than the one provided in the Java programming language could be easily implemented without modifying any other code in the program. OurRandom has two modes that can be changed during execution, if desired. The first mode, "random," generates pseudo-random numbers according to the distribution parameters provided. The second mode, "expected value," generates a number that is equal to the expected value of the distribution with the specified parameters. While running many experiments during planning, the random mode should be used. While running the simulation in parallel with the real operation, the expected-value mode should be used. Expected-value mode should be used, because external clients should not decide that the plan is going awry

due to a series of unlucky "die rolls." Expected-value mode, like the lack of reasoning in entities, is used as a variance-reduction technique.

### *7.6 Operations Monitor Structure*

The heart of the proposed methodology is the use of software agents, or Operations Monitors, to compare what is happening in the real operation with what was supposed to happen according to the plan. As described previously, the Operations Monitors exist in a tree-like hierarchy. The Top Level Monitor (described below) is the first to be created. All subsequent monitors are created as children of another monitor. All of the monitors in the prototype implementation are threads running in the same virtual machine. OMs communicate with child OMs by calling methods and with parents by calling the parent's report() method.

Each monitor is responsible for a very narrow domain. Previously the basic policy for determining the default initial set of monitors was described. This was based on the unit's mission. The monitors that could be implemented were based on the kinds of information that the prototype OpSim could provide. The seize mission described earlier and the current state of OpSim determined which Operations Monitors to implement. Figure 7 shows which monitors were implemented.

All OMs inherit from the BasicMonitor class. BasicMonitor handles the creation, management, and deletion of child monitors. Each BasicMonitor has a subscription connection to the real and simulated worlds. The BasicMonitor class manages the creation of these connections as well as the passage of queries, subscriptions, and answers. The BasicMonitor class manages the movement of messages up and down the tree of OMs. Finally the BasicMonitor class contains the basic control loop of any OM. Periodically OMs wake, do some work, make some decisions, and go back to sleep.

At the first level of the dynamic tree structure of OMs, the level that corresponds to Battlefield Functions [33], the Maneuver, Command and Control, and Intelligence monitors were implemented. Since OpSim could not simulate the Air Defense, Logistics, Fire Support, or Engineering Battlefield Functions, these OMs are merely stubs. The Maneuver, Command and Control, and Intelligence OMs report to the Top Level Monitor. The Top Level Monitor computes a utility function based on the reports and decides whether to take some action(s). Several of the more interesting OMs are discussed below.

#### *7.6.1 How the Results Monitor Works*

The Results Monitor is designed to compare the performance of types of entities in the simulation with their real-world performance. Recall that the performance of entities in OpSim is described by some probability distribution (e.g., Normal or Exponential). The results of various combat actions by entities are archived by class. M-1 equipped tank platoons have one set of results; T-72 equipped platoons have a second set; and so on. On the "real" side, the assumption is that outcomes of combats can be used to extract the various "die rolls" that would have produced those results. The Results Monitor then compares the two sets of data for each entity type.

Both Chi-Square and Kolmogorov-Smirnov (KS) goodness-of-fit tests were implemented. The KS test is the default. If the KS test determines that the distribution described by the simulation's results does not match those of the real operation (within a 95% confidence), the Results Monitor attempts to determine the correct distribution. It begins by using Maximum Likelihood Estimators (MLE) [31] to determine probability

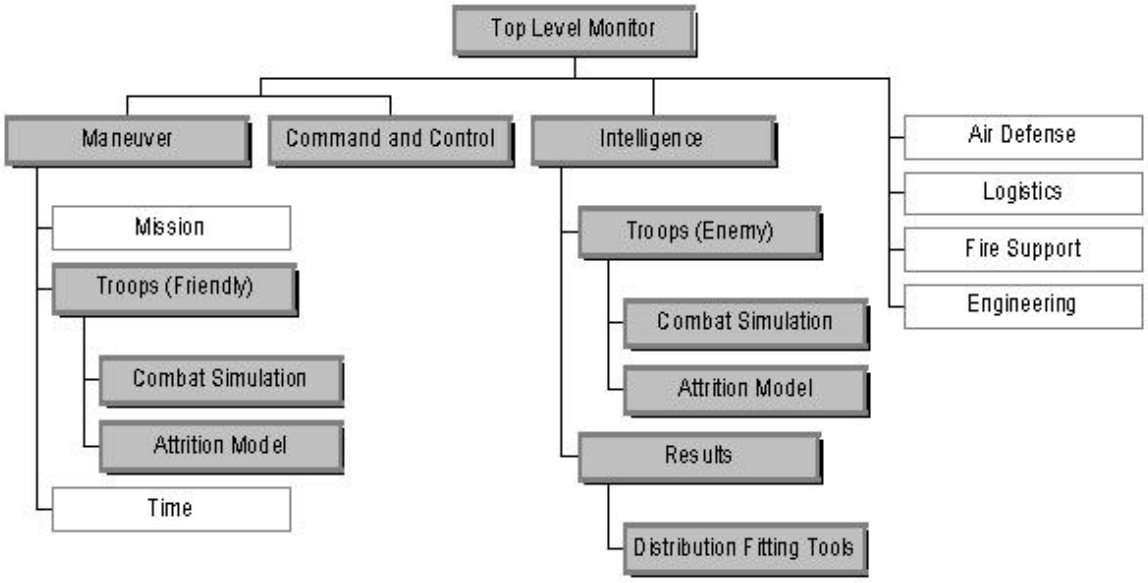


**Figure 7**. Implemented operations monitors (in dark boxes)

distributions which might match the real data. Using the parameters determined by MLE, sets of pseudo-random numbers are generated for these various distributions. The KS test is then used to determine which distribution most closely fits the real data. The closest distribution is recommended. If the closest distribution is within the same family as the current simulation distribution (e.g., Normal) this recommendation is sent to the Results Monitor's parent as a CONCERN message. If the recommended distribution is from a different family, this recommendation is wrapped in a WARNING message.

### *7.6.2 How the Forces Monitor Works*

A Forces Monitor is used to compare the strength of a designated force in the plan with the strength of that same force in the real operation. A Forces Monitor only looks at one force (side). When a Forces Monitor is created, the OM that created it tells the Forces Monitor which side to watch. In the test scenarios, only two sides were used, so there are generally two Forces Monitors: a Red Forces Monitor (enemy) and a Blue Forces Monitor (friendly). Regardless of which force the Forces Monitor is watching, it performs the same actions in the same manner.

The Forces Monitor has the most sophisticated reasoning mechanisms of any of the prototype monitors. It uses three fuzzy rules bases as well as a crisp rule base. The fuzzy rule bases are used to classify differences, and the crisp rule base is used to control the actions of the monitor. The fuzzy rule bases were built with the MATLAB Fuzzy Toolbox. MATLAB facilitates the rapid creation, display, and editing of fuzzy rules. The output from MATLAB is an FIS file. A Java implementation of the Simple Adaptive Method (SAM) of resolving fuzzy rules was created as part of the OpSim development effort [29]. This implementation of SAM works for trapezoidal, triangular, and Gaussian membership functions. When the Forces Monitor is created, it makes three FuzzyDatabase objects and loads each one with the information from an FIS file.

The first two FIS files are used to classify the differences between the real operation and the plan. The Forces Monitor computes the number of infantry units, tank units, etc. These computations take into account the strength of the units and their combat equivalency value (CEV). Factors used by the fuzzy rules to determine the criticality of any differences are the percent change from the plan to the real operation, the actual number in the simulation, the actual number in the real operation, and the importance of that type of platoon to the overall force. For instance, losing one tank platoon does not sound like much unless there was only one tank platoon. Also a small percentage change in the number of infantry platoons lost might mean several hundred soldiers if there were many infantry platoons. The output from the two fuzzy rule bases is a measure of the criticality of the difference between the real operation and the plan. If this criticality exceeds some threshold, the Forces Monitor chooses to execute either a Simulation Monitor or an Attrition Monitor.

When either the Simulation Monitor or the Attrition Monitor finishes, the third fuzzy rule base is used to categorize the difference between the end state casualties in the plan and those computed by a child Simulation Monitor or Attrition Monitor. This categorization is done by equipment type (e.g., personnel, armored personnel carriers, tanks, etc.). The fuzzy rule base uses the percentage of losses and the percentage change in losses to make this determination. For instance, the change in the real operation might double the number of armored personnel carriers lost, but these doubled losses might still be a small percentage of the total number available.

The analyses done by all three fuzzy rule bases assume that the losses for the currently executing friendly course of action versus the currently executing enemy course of action were acceptable during planning. If this were not true, why would the commander and staff choose that course of action? For this reason, the fuzzy rule bases always compare new estimates (such as personnel losses predicted at end state) with those determined during the planning process. The losses estimated during the planning process were summarized in the statistics table, shown in Figures 5 and 6.

The crisp rule base in the Forces Monitor was written in the CLIPS language [23]. JESS (the Java Expert System Shell) was used to execute the rules. JESS is a Java implementation of CLIPS with significantly greater flexibility and power. Only the basic functions of JESS were used in the Forces Monitors. The Forces Monitor creates a Rete object, and the Rete object reads the CLIPS rules from a file. The Forces Monitor uses the Rete API to assert facts into the rule base, control execution of the rules, and pull information of the Rete object through Store and Fetch operations. When the JESS rules want the Forces Monitor to report some information, the report is placed in a REPORT variable by JESS using a STORE operation. Periodically the Forces Monitor polls the REPORT variable using a FETCH operation. When the Forces Monitor finds something in the REPORT variable, that report is sent to the Forces Monitor's parent.

### *7.6.3 How the Top Level Monitor Works*

The Top Level Monitor collects the reports from all of its child OMs and makes decisions about whether to advise the commander of problems. The Top Level Monitor has three status levels. The Good status indicates that the operation is going better than the plan or only slightly worse than the plan. The Concern status indicates that conditions in the real operation are worse than in the plan (or will be at end state), but the overall success of the mission is not at risk. The Warn-

ing status indicates that the success of the mission is at risk. When the Top Level Monitor is in Concern or Warning status, the Top Level Monitor creates a window on the screen which describes all of the factors that contributed to this adverse status. In Good status, no reports are made.

The Top Level Monitor determines its state by computing a utility function based on any reports it has received. Good reports have positive values, and bad reports have negative values. When the value of the function drops below a definable threshold, the Top Level Monitor is in Concern status. When the value of the function drops below a second threshold, the Top Level Monitor is in Warning status.

The number that is added to the utility function is based on the severity of the bad news. Recall that lower-level OMs attach Warning and Concern tags to their adverse reports. This information is used to determine how bad the bad news is. The Top Level Monitor determines the value associated with good and bad reports as it adds them to the utility calculation.

The Top Level Monitor also decides whether suggestions provided by a Results Monitor require user intervention. The decision is based on the type of update suggested by the Results Monitor. If the Results Monitor suggests that the family of distribution describing some entity is correct, but the parameters need to be adjusted, the Top Level Monitor just tells the simulation to make the changes. If on the other hand the Results Monitor suggests that family of distribution must be changed, the Top Level Monitor asks the user to confirm the decision. Many families of distributions are commonly used for certain types of problems. It might not make sense to represent the performance of an entity as a Weibull or Gamma distribution.

The implementation of the Top Level Monitor was intentionally kept simple. This simplicity was designed to show that useful behavior could be accomplished without large overhead. Since many of the lower-level monitors make inferences and decisions within their narrow domain, the Top Level Monitor's work is simpler. Eventually a more intricate and more robust reasoning mechanism would be needed. Some inferences about the risk posed by the current state of the operation on the eventual outcome are probably more difficult. It seems that some bad news should be added to the utility function, but sometimes the effect of two pieces of bad news is greater than their sums. This more sophisticated reasoning system has not yet been designed or implemented.

For purposes of this prototype, the Top Level Monitor also owns the GUI associated with the dynamic tree of monitors. From this GUI, the user can manually launch other monitors or kill monitors that are running. Killing OMs from which parent OMs are waiting for reports can cause the parent OMs to block indefinitely, so deleting monitors must be done with care. More importantly, the GUI provides a tree-like depiction of

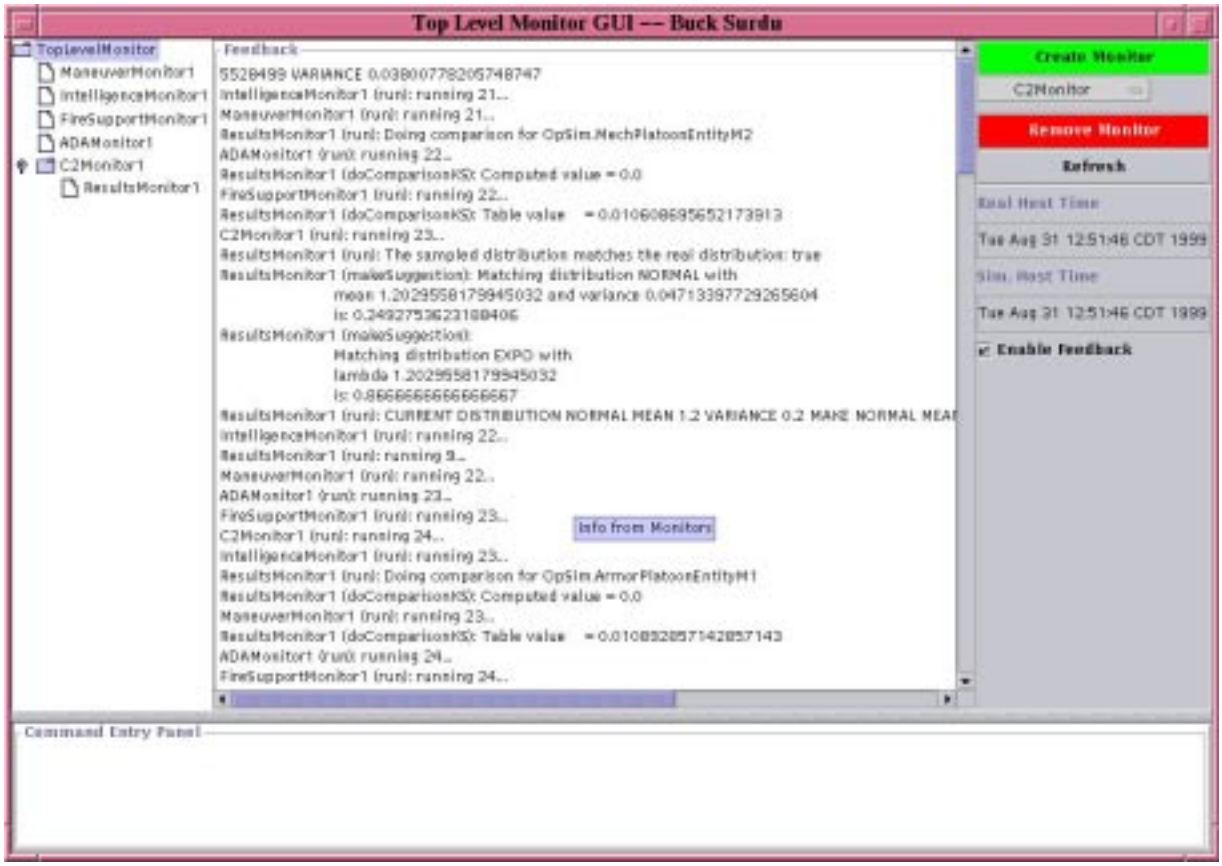


**Figure 8.** The Top Level Monitor GUI

the currently running OMs. In addition, the text window, shown in Figure 8, provides an area for the Top Level Monitor to report information to the user.

## 8. Results

### *8.1 Comparison to Other Systems*

There are no existing systems with these capabilities. While there are numerous Department of Defense and Department of the Army combat simulations, those simulations are unsuitable for this application for a number of reasons: large exercise set up time and cost, large numbers of workstations required, and the large number of personnel required. For these reasons the prototype OpSim was developed. The prototype OpSim incorporated all the capabilities needed in an operationally-focused simulation [13].

Given the existence of an appropriate simulation, however, there are still no systems designed to run a simulation of the operation in parallel with the real operation in the military domain. While Davis and his collaborators have applied a similar technique to flexible manufacturing systems [42-46], it has not previously been applied to military operations. All of the analysis of how well a plan is working is still done by humans. Since there are no existing systems with which to compare the methodology proposed in this research, an alternate method of evaluation was used.

### *8.2 Experiments*

Once the prototype built to demonstrate the feasibility of this methodology was completed, several experiments were conducted with the purpose of verification and validation. Since the WorldView portion of the methodology is still an open research issue, a second copy of OpSim was used to represent the real operation. The parameters and/or entities in the "real" OpSim were modified in specific, controlled ways to perform the various experiments.

In all these experiments, the attacking force consisted of a friendly brigade, and the defending force consisted of an enemy battalion. The entities were a mix of infantry, mechanized infantry, and tank platoons. Company headquarters elements were represented in the scenarios.

The first set of experiments involved testing the adaptive nature of the overall simulation. In the "real" OpSim, the parameters that described the probability distributions for the various entities were changed. In

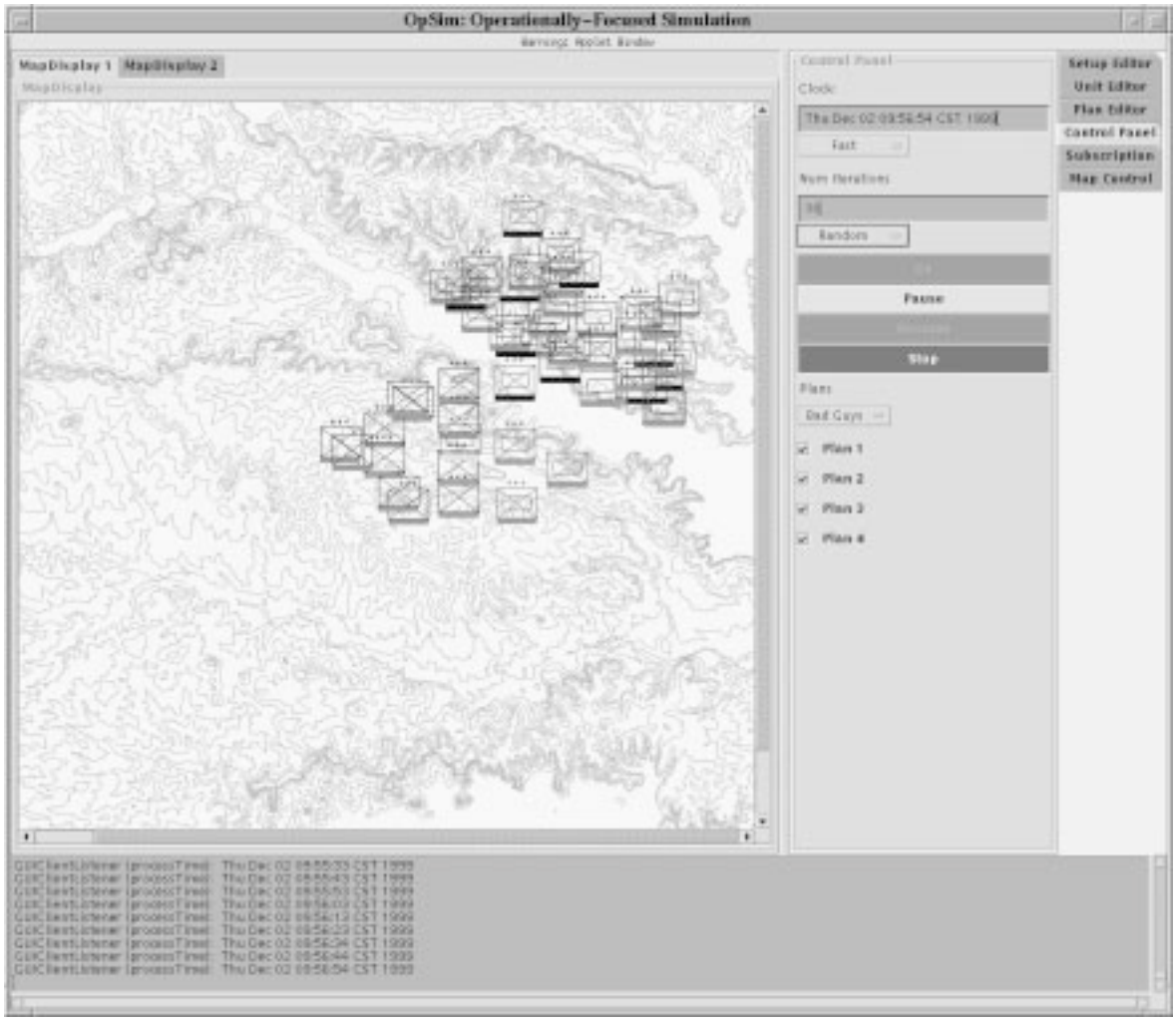


**Figure 9**. Partially completed attack by a friendly brigade against an enemy battalion

the first experiment the family of distribution remained the same, but the parameters were changed. Over time, the Results Monitor detected the differences in performance and recommended that the parameters in the "simulated" OpSim be changed to ones closer to those in the "real" OpSim. In the second experiment, the family of distribution was changed. As designed, the Results Monitor reported a recommended change of distribution family, and the Top Level Monitor created a dialog box to ask the user to confirm this change.

The second set of experiments was designed to ensure that there were no false reports of problems. Three scenarios were run in which the "simulated" OpSim and the "real" OpSim ran the same plans with the same parameters. The two simulations were both run in random mode, so that there were slight differences between the outcomes of the two simulations, but the differences were small. As expected, the Operations Monitors did not report any concerns or warnings.

The third set of experiments was designed to stimulate the Operations Monitors. This was done by adding additional enemy entities to the "real" OpSim that were not present in the "simulated" OpSim. At first a very large number of enemy entities was added, with the intention of creating a situation that clearly indicated the plan was at risk. The Forces Monitors very quickly detected the real operation diverging from the plan and launched simulations to explore the differences. The Simulation Monitors reported heavy losses, which were interpreted by the fuzzy rule bases as significant. Finally, the Top Level Monitor, collecting the reports from its children, opened a dialog box to report the warning to the commander.

The number of additional enemy entities in the "real" operation was slowly reduced to just two tank platoons. (This scenario is shown in Figures 9 and 10.) When the two currently serving Army officers looked at the scenario, they were unable to definitively classify the threat as significant or insignificant. As the battle progressed, the light infantry units in the south took heavier casualties in the real operation than in the plan. The Blue Forces Monitor launched a Simulation Monitor to explore the differences. The "real" OpSim was run in random mode, so the outcome of each battle was different. If the enemy was doing particularly well (good die rolls, so to speak) in a particular experiment, the monitors would inform the commander that there was a concern. With just two additional tank platoons

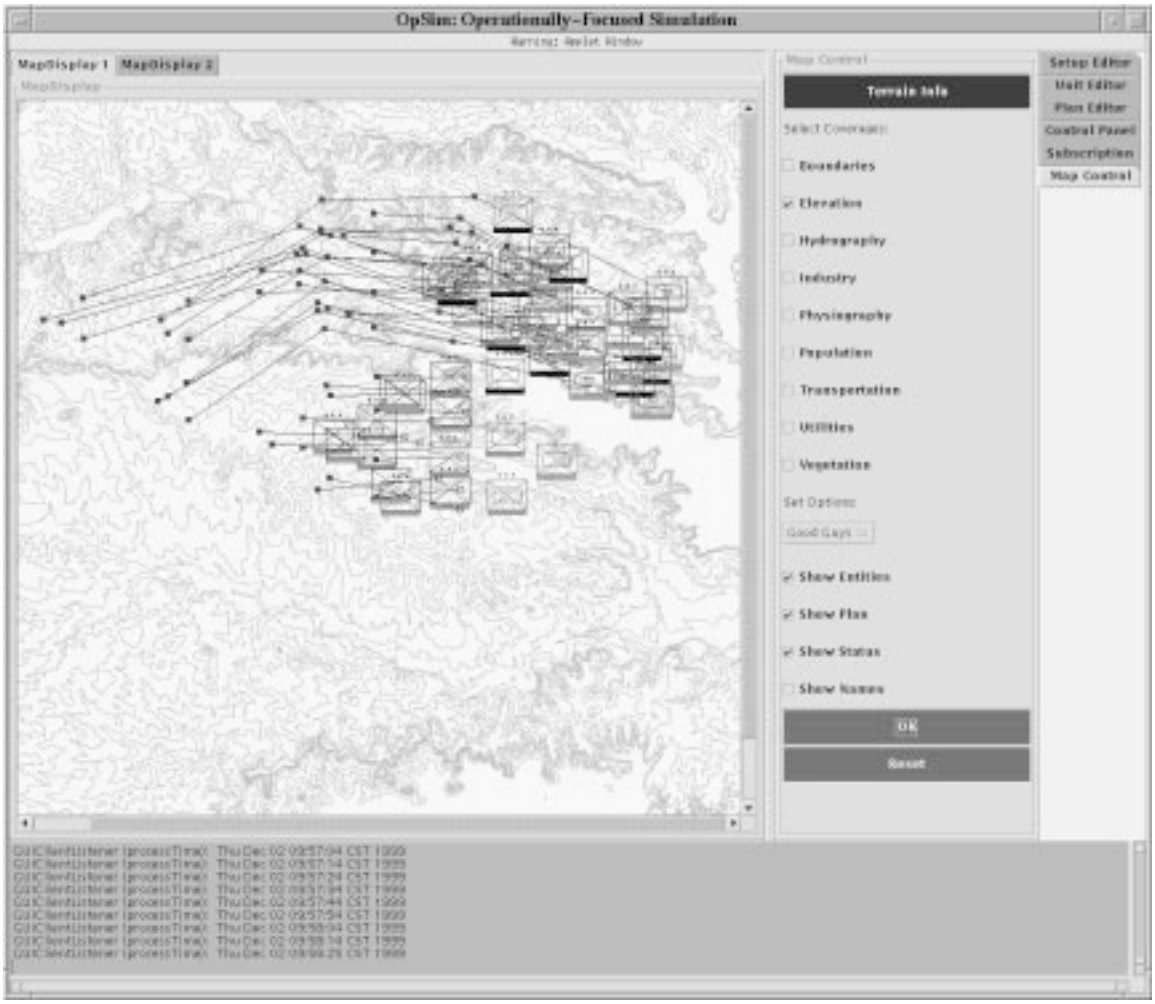


**Figure 10**. Partially completed attack with two extra enemy tank platoons and routes of march shown

the predicted outcome did not warrant a warning, just a concern.

This last experiment is the most interesting, as it provides the best demonstration of the feasibility of the proposed methodology. The human experts were unsure of the impact of the additional two enemy platoons. Even after casualties were reported, and the friendly infantry units were taking losses, the significance of the losses was unclear. The immediate reaction on seeing two light infantry platoons jump from green to red status was that the plan was at risk. While the increase in casualties was alarming, it had only a moderate impact on the outcome of the operation. The friendly forces of course lost more personnel and equipment, but they reached their eventual objective in strength.

## 9. Future Work

There are no similar environments with which to compare this methodology. While there are a number of training simulations that could be used in the COA development and analysis described in this paper, none has been applied for use during operations, and there are no multi-agent analysis tools in this domain. While a number of experiments were conducted that indicate that this proposed methodology is useful to commanders and their staffs, more work is needed before this assertion can be demonstrated in a statistically significant way. More missions need to be decomposed to determine the automatic (default) set of OMs to create (as discussed in Section 6). A variety of new scenarios need to be developed which test the various aspects of the methodology and rigorously validate the inference mechanisms of each of the OMs. As it becomes clear what additional information is needed by the OMs, more hooks need to be created into OpSim to get that information. As stated earlier, one of the contributions of this work is to determine what information an operationally-focused simulation needs to be able to provide.

The prototype OpSim and Operations Monitors were never intended to be used in the field. They were intended to demonstrate the feasibility of the overall methodology. The creation of a fieldable combat simulation is a difficult task, often requiring teams of programmers several years. Similarly, the inference mechanisms associated with each of the various Operations Monitors will probably become quite complex by the time a fieldable system is developed. Each Operations Monitor might require its own research and development effort. This research has demonstrated the usefulness of a fielded system with the capabilities described (and implemented to prototype levels). A large amount of work is needed before such a system could be given to war fighters.

The prototype OpSim provides some sophisticated and useful information to commanders and planners. Before OpSim would be ready for field use, many more capabilities are needed. It is unclear whether it would be easier to trim down an existing Army simulation to just those components needed for an operationally-focused simulation and then add needed capabilities, or whether it would be easier to build a new operationally-focused simulation. Assuming that one would proceed by enhancing OpSim to a fieldable state, below is a short list of needed enhancements.

United States military forces are increasingly deployed as part of coalition operations. The prototype OpSim has the capability to represent multiple forces; however, any force not one's own is considered to be an enemy. OpSim should be enhanced so that forces can have relationships other than enemy, such as neutral, friendly, etc. These relationships should be represented in such a way that the relationship need not be symmetrical. For instance, side A might think that side B is neutral, but side B might consider side A to be an enemy. This enhancement would make OpSim appropriate for a wide spectrum of military operations, not just two-sided conventional fights.

The prototype OpSim only simulates direct-fire ground engagements. Modern military operations include aircraft, anti-aircraft weapons, indirect fire weapons (e.g., artillery and mortars), precision-guided munitions (including sensor-to-shooter links), mine fields, chemical weapons, and obstacles. These aspects of modern warfare must be represented in OpSim in order to make it useful in the field. In addition, tactical communications networks, both voice and data, must be represented, because the loss or damage of these networks has a significant impact on the outcome of operations.

German Field Marshal Irwin Rommel once said that amateurs study tactics and professionals study logistics. Often the outcome of an operation depends as much on whether fuel, ammunition, and food can be delivered to the soldiers as it does on the results of individual combats. Logistics should be represented within the simulation to provide better information to the commander and staff. During planning, the representation of logistics would help the logistics officers determine the number and location of re-supply points, supply routes, etc. During the execution of the operation, a logistics Operations Monitor could monitor the number of tank rounds remaining in a platoon, for instance, versus the planned number. If the tank unit appeared to be using ammunition at a faster-than-planned rate, a Simulation Monitor could be used to determine if that unit will run out of ammunition prior to the end of the operation.

One of the stated objectives of the proposed methodology is to decrease the amount of information that planners and commanders have to process. While the task of identifying when the plan is at risk is not trivial, a much more difficult problem is for the system to then recommend a solution. This relates to the course of action generation research conducted by Fiebig, Hayes,

and others [11, 12, 47]. Solving this problem requires the creation of a detailed and complete language for describing courses of action as well as the development of techniques for generating them. Courses of action are represented as plans in OpSim. It is unclear whether this representation would be useful in a tool designed to invent new courses of action for an operation. This course of action generation capability would be useful during planning. It would also be very useful during the conduct of the operation. After the Top Level Monitor identified that the plan was at risk, it could present the commander with some number of recommended alternative solutions.

The principle that determines which OMs are automatically launched requires a mapping from mission tasks to a default set of OMs. This was only done for two tasks, seize and defend in sector, during the construction of the prototype system. The Army's Maneuver Control System (part of the ABCS [15]) contains an operations order generation tool. This tool lists all the mission tasks. Each of these tasks should be analyzed to create a default set of OMs for each task.

Only a small number of OMs was developed as part of this prototype. The first step would be to flesh out the OMs that are merely stubs, such as Fire Support, Logistics, Engineering, and Air Defense in the first level of the hierarchy. (This clearly requires that these Battlefield Functions be represented in OpSim.) In the prototype hierarchy, the time and mission monitors under the Maneuver Monitor were not implemented. (In most cases what information OpSim was capable of providing determined which OMs were not implemented.)

Each OM could be the subject of its own research project to determine all the information needed to make inferences about the plan versus the real operation. As part of this research, the appropriate reasoning mechanism should be identified. For example, the Top Level Monitor adds all evidence together to determine whether the operation is likely to succeed. There may be instances in which the combination of two pieces of evidence is greater than the sums of the parts. The OMs must be robust enough to deal effectively with uncertainty while making quick, accurate assessments of the situation.

## 10. Summary

This paper proposes a methodology for using simulations during an ongoing operation. This includes the use of an operationally-focused simulation that runs in real time, simulating the plan. This methodology also includes the use of intelligent agents, Operations Monitors, to compare the events in the real operation versus those in the plan. These agents query both the representations of the real operation and the simulation to find deviations from the plan. The agents then launch various tools to determine the effects of these deviations. If the effects are significant, the agents advise the commander and staff.

There is wide recognition within the Army and Department of Defense that a system to facilitate rapid course of action analysis is needed. The design of the overall methodology and the requirements for the operationally-focused simulation support the use of OpSim in this role. In addition, because OpSim supports the querying of information and subscribing to information, it makes possible the use of software agents to help monitor the operation. The technology used to pass this information is less important than the existence of this capability. OpSim uses message passing; however, CORBA, Persistent Object Protocol [6], Java RMI, or some future technology could be substituted. The use of ASCII-based messages across TCP/IP sockets was designed to allow the easy creation of Operations Monitors without specifying an implementation language.

OpSim provides feedback to a planner. OpSim estimates expected losses when executing various friendly courses of action versus various enemy courses of action. This estimation can be done before an operation or during the operation. It also provides some feedback about how long it will take to accomplish the mission. These are pieces of information often desired by planners [41, 48]. Given an attrition model that the users can accept, OpSim provides answers that "make sense." The answers OpSim provides are necessary, if not sufficient, for planners to decide between courses of action.

## 11. References


[1] Delany, P.J., Captain, U.S. Army, Army Modeling and Simulation Office, Personal Communication, 28 October 1998.

[2] Surdu, J.R. and Pooch, U.W. "A Methodology for Applying Simulation Technologies in the Mission Operational Environment." *Proceedings of the IEEE Information Technology Conference*, Syracuse, NY, 1-3 September 1998, pp 45-48.

[3] Gunning, D. "Command Post of the Future." At *http://www.mole.dc.isx.com/cpof*, 15 October 1998.

[4] Clausewitz, C.V. *On War*, Princeton University Press, Princeton, NJ, 1984.

[5] Balash, J. "Janus." At *http://www.stricom/army.mil/PRODUCTS/JANUS*, March 1999.

[6] Rodio, S. "ModSAF." At *http://www.stricom.army.mil/STRICOM/E-DIR/ES/MODSAF*, March 1999.

[7] Sackett, D.E. "Simulations to Save Time, Money, and Lives." *Lawrence Livermore National Laboratory Science and Technology Review*, Vol. 96, No. 11, pp 4-11, November 1996.

[8] Rogers, M. "Warfighter's Simulation." At *http://www.stricom.army.mil/PRODUCTS/WARSIM*, March 1999.

[9] Surdu, J.R. and Pooch, U.W. "Connecting the Operational Environment to Simulation." *Proceedings of the Advanced Simulation Technology Conference: Military, Government, and Aerospace Simulation*, San Diego, 11-14 April 1999, pp 94-99.

[10] Wyden, P. *Bay of Pigs*, Simon and Schuster, New York, 1979.

[11] Fiebig, C.C., Hayes, C.C. and Schlaback, J. "Human-Computer Interaction Issues in a Battlefield Reasoning System." *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Orlando, FL, 12-15 October 1997, pp 3204-3209.

[12] Fiebig, C.B. and Hayes, C.C. "DAISY: A Design Methodology for Experience-Centered Planning Support Systems." *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, 11-14 October 1998, pp 920-925.

[13] Surdu, J.R., Haines, G.D. and Pooch, U.W. "OpSim: A Purpose-built Distributed Simulation for the Mission Operational Environment." *Proceedings of the International Conference on Web-Based Modeling and Simulation*, San Francisco, 17-20 January 1999, pp 69-74.

[14] Blais, C.L. and Garrabrants, W.M. "Simulation in Support of Mission Planning." *Proceedings of the Advanced Simulation Technology Conference: Military, Government, and Aerospace*, San Diego, 11-17 April 1999, pp 117-122.

[15] Bessler, J.E. "Army Battle Command System (ABCS) Capstone Requirements Document (CRD)." Tech. Rep. Revision 1.0, TPIO-ABCS, Ft. Leavenworth, KS, 10 February 1998.

[16] Maes, P. "How to Do the Right Thing." Tech. Rep. 1180, Massachusetts Institute of Technology, Cambridge, MA, January 1989.

[17] Maes, P. "Agents that Reduce Work and Information Overload." *Communications of the ACM*, Vol. 37, No. 7, pp 31-41, July 1994.

[18] Maes, P. "Situated Agents Can Have Goals." In *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, P. Maes (ed.), MIT Press, Cambridge, MA, pp 49-70, 1994.

[19] Hayes-Roth, B. "An Architecture for Adaptive Intelligent Systems." *Artificial Intelligence*, Vol. 72, No. 1, pp 329-365, January 1995.

[20] Franklin, S. and Graesser, A. "Is it an Agent, or Just a Program? A Taxonomy for Autonomous Agents." In *Intelligent Agents III: Proceedings of the Workshop on Agent Theories, Architectures, and Languages*, J. Jörg, P. Müller (eds.), Springer-Verlag, Berlin, pp 21-36, 1997.

[21] Mitchell, T.M., *Machine Learning*, McGraw-Hill, Boston, 1997.

[22] Ourston, D. and Mooney, R.J. "Theory Refinement Combining Analytical and Empirical Methods." *Artificial Intelligence*, Vol. 66, No. 2, pp 311-344, 1994.

[23] Giarratano, J. and Riley, G. *Expert Systems: Principles and Programming*, PWS-Kent Publishing, Boston, 1989.

[24] Lacher, R., Hruska, S. and Kuncicky, D. "Backpropagation Learning in Expert Networks." Tech. Rep. TR91-015, Department of Computer Science, Florida State University, Tallahassee, FL, March 1991.

[25] Lacher, R.C., Hruska, S.I. and Kuncicky, D.C. "Back-Propagation Learning in Expert Networks." *IEEE Transactions on Neural Networks*, Vol. 3, No. 1, pp 62-72, January 1992.

[26] Lacher, R.C. "Expert Networks: Paradigmatic Conflict, Technological Rapprochement." *Mind and Machines*, Vol. 3, No. 1, pp 53-71, January 1993.

[27] Kuncicky, D., Hruska, S. and Lacher, R.C. "Hybrid Systems: The Equivalence of Rule-Based Expert System and Artificial Neural Network Inference." *International Journal of Expert Systems*, Vol. 4, No. 3, pp 281-297, July-September 1992.

[28] Mitchell, T., Caruana, R., Freitag, D., McDermott, J. and Zabowski, D. "Experience with a Learning Personal Assistant." *Communications of the ACM*, Vol. 37, No. 7, pp 81-91, July 1994.

[29] Yen, J. and Lengari, R. *Fuzzy Logic: Intelligent Control and Information*, Prentice Hall, New York, 1999.

[30] U. S. Army, FM 100-6, Information Operations, Washington, DC, Headquarters, Department of the Army, 1996.

[31] Milton, J.S. and Arnold, J.C. *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences*, McGraw-Hill, New York, 1995.

[32] Pooch, U.W. and Wall, J.A. *Discrete Event Simulation: A Practical Approach*, CRC Press, Boca Raton, FL, 1993.

[33] U. S. Army, FM 100-5, Operations, Washington, DC, Headquarters, Department of the Army, 1993.

[34] National Imagery and Mapping Agency, "NIMA Public Web Page." At *http://www.nima.mil*, December 1999.

[35] Dept. of Defense, MIL-STD-2407, Department of Defense Interface Standard for Vector Product Format, Fairfax, VA, National Imagery and Mapping Agency, 1996.

[36] National Imagery and Mapping Agency, "VPF Overview Web Page." At *http://164.214.2.59/vpfproto/index.htm*, December 1999.

[37] Dept. of Defense, MIL-V-89032, Military Specification, Vector Smart Map (VMap) Level 2 (Draft), Fairfax, VA, National Imagery and Mapping Agency, 1996.

[38] DMAS Office. "SEDRIS Web Page." At *http://www.dmso.mil/portals/sedris.html*, December 1999.

[39] Dupuy, T.N. *Numbers, Predictions, and War: Using History to Evaluate Combat Factors and Predict the Outcome of Battles*, The Bobbs-Merrill Company, Indianapolis, IN, 1979.

[40] Dupuy, T.N. *Attrition: Forecasting Battle Casualties and Equipment Losses in Modern War*, Nova Publications, Falls Church, VA, 1995.

[41] U.S. Army, FM 101-5, Staff Organization and Operations, Washington, DC, Headquarters, Department of the Army, 1998.

[42] Davis, W.J. "Developing an Intelligent Controller for Integrated Manufacturing Resource Planning." Tech. Rep. DRAFT, University of Illinois at Urbana-Champaign, Urbana, IL, October 1998.

[43] Davis, W.J. "A Framework for the Distributed Intelligent Control of Advanced Manufacturing Systems." Tech. Rep. DRAFT, University of Illinois at Urbana-Champaign, Urbana, IL, October 1998.

[44] Davis, W.J. "On-Line Simulation: Need and Evolving Research Requirements." In *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, J. Banks (ed.), John Wiley and Sons, New York, 1998, pp 465-516.

[45] Davis, W.J. "On-Line Simulation for Torpedo Avoidance." At *http://www-msl.ge.uiuc.edu/~brook/boat*, November 1998.

[46] Davis, W.J., Chen, X., Brook, A. and Awad, F.A. "Implementing On-line Simulation with the World Wide Web." *SIMULATION*, Vol. 73, No. 1, pp 40-53, January 1998.

[47] Hayes, C.C., Ergan, H., Tu, N., Liang, M., Jones, P., Barger, R., and Wilkins, D. "CoRAVEN: Intelligent Tools to Provide Multi-Perspective Decision Support and Data Visualization." Tech. Rep. UMN-IE-99-001, University of Minnesota, Minneapolis, MN, 14 December 1999.

[48] U. S. Army, ST 100-9, Techniques and Procedures for Tactical Decisionmaking, Fort Leavenworth, KS, U.S. Army Command and General Staff College, 1991.

TECHNICAL ARTICLE

*SIMULATION* 74:3, ##-##
©2000, Simulation Councils, Inc.
ISSN 0037-5497/00
Printed in the United States of America

# Creating Galaxies on a PC

**Stanislaw Raczynski**
McLeod Institute for Simulation Sciences
Panamericana University
489 Augusto Rodin, 03910 Mexico City, Mexico
Web: http://www.raczynski.com
E-mail: stanracz@netservice.com.mx

*It is shown that galactic simulation does not require a supercomputer. The capacity of new PCs is enough to simulate an N-body system with the number of bodies sufficient to see a galaxy formation from chaos to a disk with a spiral galactic structure. Naturally, simulations with millions of bodies realized on parallel supercomputers provide more realistic images. However, the number of bodies is not a crucial factor if rather qualitative results are expected. By simulating a pure gravitational system with only 500 particles on a PC, one can observe a fascinating history of a galaxy evolution. Some algorithms for N-body integration are discussed and an implementation of a simple "neighbor and far interactions" algorithm is presented. The complete history of a simulated galaxy is shown, including the initial collapse, expansion, formation of the early cloud with rotational movement and, finally, a spiral structure.*

**Keywords**: Galactic simulation, N-body systems

## 1. Introduction

There is a huge body of literature and a large number of numerical methods for the N-body problem. We do not pretend to give any exhaustive discussion here (it would, rather, need a book), and only some recent results will be mentioned. In fact, it is enough to type the search phrase "galactic simulation" to get many references and good pictures from the Internet. More pages can be found with the phrase "N-body," though not all of them refer to galactic dynamics.

To solve the N-body problem we must integrate a system of second order ordinary differential equations of the form:

$$dx_i^2 / dt^2 = f_i(x) \text{ for } i = 1,2,....,N,$$

where $f_i(x)$ is the total force, $x = (x_1, x_2, x_3, ..., x_N)$, $x_i \in R^3$. Here $f_i$ is the vectorial sum of all the forces the body receives from the other bodies. This force is proportional to $1/d^2_{ij}$, where $d_{ij}$ is the distance from body $i$ to body $j$. Thus, the number of forces grows with $N$ square, and for big $N$, the computer time may be unacceptable. Another difficulty consists of the nonlinearity of the problem. In fact, looking at the above equation, we can see that the space where the bodies move is full of singularities, each body position being