SYNCHRONIZED SIMULATIONS IN PLANNING SYSTEMS

John M. D. Hill
Department of Computer Science
Texas A&M University
College Station, TX 77843, USA
hillj@cs.tamu.edu

John R. Surdu
Information Technology Operations Center
United States Military Academy
West Point, NY 10996, USA
john-surdu@usma.edu

James Vaglia
Department of Computer Science
Texas A&M University
College Station, TX 77843, USA
jvaglia@cs.tamu.edu

Udo W. Pooch
Department of Computer Science
Texas A&M University
College Station, TX 77843, USA
pooch@cs.tamu.edu

KEYWORDS

Military, Planning Aids, Decision Making

ABSTRACT

The U.S. Military is embracing technologies that provide enhanced situational awareness and enable rapid decision-making. The idea is to gain option dominance over an adversary through superior collection, integration, and presentation of information and more rapid decision-making. Considerable effort is being expended to meet the current challenge of developing systems to analyze, filter, and integrate the relevant information into a common operational picture. Once commanders have this improved ability to observe and understand the battlefield environment, the next challenge will be to determine the best ways to make use of that knowledge. Meeting that challenge with appropriate planning and decision support systems should not have to wait until the information-providing systems are operational. One way to develop the planning systems in parallel with the information-providing systems is to produce that information as if the systems already existed. One technique for exercising these planning systems is to run simultaneous simulations – one that provides the 'real world' state and stimulates the system, and one or more simulations that the planning system uses to track progress through the plan and develop new branches based on the actual state. For this technique to work, the simulations must remain synchronized as the operation progresses. This paper explains the motivation for developing such synchronized simulation systems, determines the associated requirements, and describes two planning systems that use synchronized simulations to evaluate planning methodologies.

BACKGROUND

There are many systems already in use that provide a wealth of information about enemy and friendly activities, the weather, the terrain, etc. – and more systems are under development. Now the challenge is to integrate all of the sensor data and reports into a common operational picture. The integration and presentation of battlefield information is an area of very active research, and it is reasonable to expect high-fidelity, high-value systems in the near future.

The remaining question is how best to use the information that will be provided by these future systems. One approach to answering the question is to replicate that information as if the systems already existed, and start building the planning systems in parallel with the information-providing systems. Planning systems have to maintain a representation of the 'real world' and the attributes and activities of the entities therein. A merged planning and execution monitoring system must compare the current state (and perhaps projected states) against the plan and make a determination whether re-planning is necessary.

One technique for exercising these planning systems is to run simultaneous simulations – one that provides the 'real world' state and stimulates the system, and one (maybe several instances) that the planning system uses to track progress through the plan and develop new branches based on the actual state. The multiple simulations must remain synchronized as the operation continues. If the start states of the various planning simulations are not “close” to the real operation, any conclusions or recommendations made by the simulation would be suspect.

For purposes of this paper, simulation synchronization is defined as the process of keeping the state of the simulation (including the simulation clock) within some small amount of variance of an external system, such as another simulation, a real-world system, a command and control system, etc. This does not mean that the simulation must be running in real time. The difference between the simulation state and the system to which it is synchronized should be small (or at least constrained by some small tolerance).

RELATED WORK

Gilmer and Sullivan have been experimenting with multi-trajectory simulations and recursive simulations [1]. Multi-trajectory simulation involves running multiple simulations down all branches simultaneously, rolling the results back to the original fork, and using some measure of utility to determine which branch in the plan has the greatest potential for success. For instance, whenever the simulation reaches a decision point, a new simulation is created for each possible branch. When end states (i.e., leaf nodes) or the simulations have run for a specified amount of time, they terminate themselves and report some statistics to the simulation that created them. As an extension of this work, Gilmer and Sullivan built a capability they refer to as recursive simulation. In this approach, whenever an entity reaches a decision point, the entity launches a multi-trajectory simulation to determine which path to take. This is similar to what Davis does with on-line simulation and, in particular, his boat-and-torpedo simulation [2, 3].

A research issue described by Collier [4] is the automated generation of courses of action for the system to consider. Fiebig, Hayes, and Schlabach [5] described a system for generating many courses of action in the military domain using genetic algorithms. Each generation of potential COAs was evaluated by a coarse, low-fidelity simulation. They also used a scheme to ensure that new courses of action generated through crossover and mutation were in fact different from the existing courses of action.

A logical extension of using simulations to generate courses of action for humans is generating courses of action for simulation entities. Porto, Fogel and Fogel [6] developed a prototype system that used Genetic Algorithms [7] to create plans for simulation entities. Similarly, Mason and Moffat used a number of artificial intelligence techniques to perform command and control functions within a simulation, namely data fusion, decision-making and planning, and plan supervision [8]. In both cases, the simulations' synchronization was relatively straightforward, since the simulations needed only be seeded with the current state of the simulated battle.

Davis has been researching the use of simulations during the operation of manufacturing system [2, 9, 10]. Andersson and Olsson proposed a very similar use of simulation during the operation of a customer order driven assembly line [11]. The use of simulation during military operations was echoed recently by Ferren: "When much-faster-than-real-time simulation becomes practical, the warfighter would have a predictive tool available, online in the field, and integral to their weapons, mobility, and communication systems." [12]

Davis also discussed a concept called *autovalidation* that compares the result of an on-line simulation over time against the real operation and updates the simulation to improve its accuracy [2]. Davis admitted that there are no theoretically sound methods of performing autovalidation and asserted that this is an open research issue. Surdu and Pooch discussed the need for such a feedback mechanism and one proposed approach [13]. This feedback mechanism is one of the important contributions of simulation synchronization.

SYNCHRONIZED SIMULATION REQUIREMENTS

There are many synchronization requirements in using simulations to stimulate and test a planning system. The most important requirement is to keep the simulation(s) that represent or "play" the plan in synchronization with the stimulation system that is simulating the real operation. Changes in state of the *stimulator* must propagate in a manner that keeps the systems consistent. The *stimulator* or the planning system may have to pause until the other catches up. This ensures that the planning system will also be able to stay synchronized with the fielded information-provision systems.

For testing purposes, the *stimulator* may be operating in faster-than-real-time. If the planning system uses simulations to evaluate situations and make recommendations, then those internal simulations must operate more quickly or otherwise adjust to the changed timing (such as use a reduced-capability but faster simulation). Also, those recommendations must be provided with sufficient lead-time, with respect to the time scale, to be useful to the tester or planner.

A final requirement is to incorporate the ability to update the parameters of the planning systems simulations based on feedback from the *stimulator* (or the real systems), improving their ability to predict planning performance.

ANTICIPATORY PLANNING SUPPORT SYSTEM

The traditional Military Decision Making Process (MDMP) focuses on developing a few friendly Courses of Action (COAs) against the "most-likely / most-dangerous" enemy COAs. This results in a very detailed plan that considers only a few branches. There is a well-known axiom that "No plan survives the first shot" - another way of saying that a branch has occurred during execution that was not included in the plan. In response, the commander and the staff transition into reactive mode. Yet, the military is now capable of producing unprecedented amounts of battlefield information that could be used to better anticipate the flow of the battle. Military planners need a new way to incorporate this continuous feed of battle information into a simultaneous planning and execution mechanism so that they achieve and maintain "option dominance". Brigadier General (ret.) Wass de Czege has proposed a new approach called "anticipatory planning" that merges planning and execution and replaces reaction to events with anticipation of events [17].

Hill, Surdu, and Pooch are implementing the anticipatory planning process in the *Anticipatory Planning Support System* (*APSS*) [18]. In this system, the user develops as many branches as reasonably possible in the initial planning process. As the operation progresses the plan is continuously updated based on actual events. Future branches that are known to be invalid are nominated for pruning and new branches are developed – well before they occur in execution. In this way, planning effort is expended on the most likely or most valuable branches. See Figure 1, where the inner workings of the *APSS* are depicted. The methodology underlying the APSS includes six fundamental processes, described below. For clarity, the processes are underlined and the components of the system are capitalized and italicized.

To track the actual operation, a *World Integrator* synthesizes actual or estimated information into a *World View* that provides the *Actual State* of execution.

To represent the plan, a *Plan Description* is dynamically built to manage the many tree-like branches that occur in planning and execution of an operation. *Nodes* retain the *Planned States* at specific points in the plan. *Branches* contain the entity task lists, status changes, and the task pre-conditions.

To detect plan deviations, *Execution Monitors* are attached to *Nodes* and derive *Anticipated States* by forward simulation from the *Actual State* along the appropriate *Branches*. The *Execution Monitor* compares the *Anticipated State* with the *Planned State* at that *Node*, determines the significance of any differences, and recommends that the *Planning Executive* (discussed below) conduct re-planning on a *Node* and/or prune certain *Branches* from the *Plan Description*

To conduct re-planning, *Planners* generate and evaluate new *Branches*. A *Branch Generator* uses a genetic algorithm and inference mechanisms that consider possible friendly or enemy actions and produces significant, representative, *Branches*. The *Planner* invokes a *Branch Evaluator* to examine a *Branch* using simulation and inference mechanisms, then determine viability measures and outcome confidences.

To control and prioritize monitoring and re-planning, a *Planning Executive* centralizes the assignment of *Execution Monitors* and *Planners* to *Nodes* with the time and system resources available. The *Planning Executive* prioritizes planning to maintain a balance between anticipating as many future branches to the plan as possible and constraining the planning effort.

To support the planning system with simulations, several discrete event simulation (DES) mechanisms are used. As the user constructs plans DES is used to determine the results of entity interaction and ensure the constructed plan is valid. Similarly, a DES is used when Branches are created by the *Branches Generator* to determine a new *Planned State* at the conclusion of the *Branch*. A DES is also used in a 'playback' mode to determine and display the actions taken within *Branches* by building an event list for the display and executing it in accordance with a user-selected time scale. Finally, the *Execution Monitors* use a DES to produce *Anticipated States* for comparison with *Planned States*.

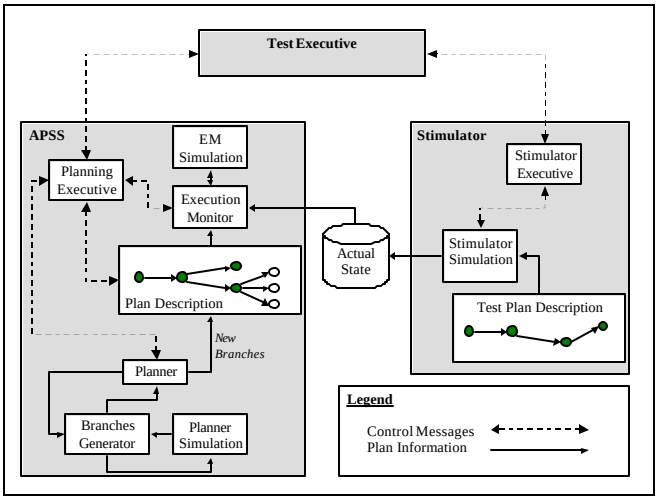

Figure 1: Testing of the APSS with a Stimulator

SYNCHRONIZATION ISSUES IN APSS TESTING

In a real military operation, automated command and control assets would provide the *Actual State* of the operation. For the purposes of this research, the *Actual State* is produced by an external mechanism that represents the activities of the *World Integrator* and *World View* components. See Figure 1 for a depiction of the test setup. The *Test Executive* provides the interface for the human tester to load the appropriate *Plan Description* into the *APSS* and the modified *Plan Description* into the *Stimulator*. Also, the *Test Executive* allows the tester to establish the time scale for the operation, and to start, pause, and stop the operation. The *Stimulator* operates on a test *Plan Description* that contains controlled differences from the *Plan Description* in the *APSS*. The *Stimulator* processes the test *Plan Description* to periodically produce *Actual States*. Meanwhile, the *APSS* is processing the *Actual States* on its own schedule. Unfortunately, this causes some timing concerns. For example, if the *APSS* is operating more slowly than the *Stimulator*, it could be generating new *Branches* for *Nodes* that have already been passed in "the real world."

The monitoring and re-planning process, and its associated time concerns, is represented in Figure 2. An *Execution Monitor* has been attached to a *Node*. This *Execution Monitor* periodically creates an *Anticipated State* by simulating forward from the *Actual State*. If the difference between the *Anticipated State* and the *Planned State* is enough to invalidate *Branches* out of that *Node*, the *EM* recommends re-planning to the *Planning Executive*. The time window labeled (1) represents the time used by the *Execution Monitor* in this process.

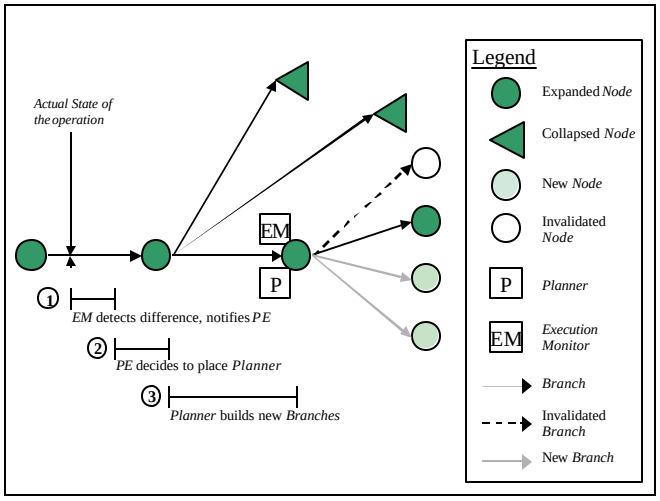

Figure 2: *APSS* Monitoring and Re-planning Process

The *Planning Executive* examines and prioritizes the recommendations from all of the *Execution Monitors*, and allocates *Planners* to the highest priority *Nodex*. The time window labeled (2) represents the amount of time between the *Planning Executive* receiving the recommendation from the *Execution Monitor* and deciding to place a *Planner* on that *Node*.

The *Planner* executes a genetic algorithm and an inference mechanism to produce some number of new, representative, *Branches*. In the worst case, the *Planner* has until the *Actual State* reaches the *Node* to produce the new *Branches*. Time window (3) represents this maximum re-planning time.

The *Test Executive* keeps the *APSS* and the *Stimulator* synchronized by sending control messages and receiving notifications from the two systems. The *Test Executive* maintains the master clock and the time scale. The *Stimulator* notifies the *Test Executive* when each new *Actual State*, containing a time stamp, is ready. At the appropriate time, the *Test Executive* sends a message to the *Stimulator* allowing it to post the new *Actual State*. The *Stimulator* replies with a confirmation of the posting, and then the *Test Executive* notifies the *Planning Executive* that a new *Actual State* is available.

Meanwhile, the *Test Executive* provides time updates to the *Planning Executive*, which uses them to control the flow of monitoring and re-planning. In chronological order by time windows, the process is as follows:

Time window (1): Ideally, the *Execution Monitors* must complete their forward simulation and decide whether to make a recommendation before the next *Actual State* is available. Depending on the time scale in use, this is not always achievable. Currently, the *Execution Monitor* ignores any new *Actual States* that arrive while it is processing. Future implementations will reduce the fidelity and resolution of the underlying simulation in an attempt to complete the processing in time.

Time window (2): The *Planning Executive* must place the *Planner* on a *Node* well enough in advance for the *Planner* to complete its work. Currently, the *Planning Executive* predicts the re-planning duration (time window (3)) by using a simple function of the genetic algorithm parameters (i.e., a constant times the number of generations times the number of genomes). A more sophisticated future system would consider historical branch generation times under various load conditions and adjust the expected duration of time window (3) appropriately.

Time window (3): If the *Planner* has not produced new *Branches* when the time update equals the time stamp of the *Planned State* in the *Node*, the *Planner* keeps working; however, the lack of a completion message from the *Planner* tells the *Planning Executive* that the system is in danger of failure. Currently, the *Planning Executive* has no mechanism for handling this impending failure. Future implementations will allow the *Planning Executive* to adjust the re-planning parameters (number of generations, etc.) and apply more *Planners* in an attempt to produce "good enough" new branches before the failure point.

If the time update passes the time stamp in the *Planned State* of the *Node*, and there are no *Branches* that accurately follow the actual operation, then the *APSS* has lost the ability to anticipate the flow of the battle. Currently, the system declares failure. Future implementations will immediately start re-planning in an attempt to catch up.

OPERATIONALLY FOCUSED SIMULATION

Surdu, Haines, and Pooch constructed a prototype system called OpSim to use simulations and software agents to conduct operations monitoring [14]. In this methodology, a simulation running in near real time simulated the plan. Software agents, called Operations Monitors or OMs, each looking at specific aspects of the plan, compared the actual operation with the planned operations. When significant differences were encountered, the system could launch other tools and simulations to determine whether these differences would have a significant impact on the desired end state of the operations. As it predates work on APSS, OpSim merely identified problems; it did not recommend solutions. In OpSim there were two major synchronization issues. The first was how to update the parameters of the simulation so that it became a better predictor of plan performance and will be discussed in greater detail in this section. The second was how to keep the simulation running the plan synchronized with the *stimulator* and was discussed in the previous two sections.

As with previous approaches, OpSim could easily seed other simulations meant to explore the ramifications of differences between the plan and the real operation. The harder issue is how to update the simulation running the plan. Once some number of agents have identified differences between the plan (simulation) and the real operation (*stimulator*) and have determined whether these differences matter it would be counterproductive for the simulation to continue with invalid state information.

In order for OMs to adequately compare the real operation with the simulated operation, the two representations must be "close." If the simulated plan is allowed to continue to diverge from the real operation over time they may become almost completely unrelated. Any analysis the OMs would perform at that point would be meaningless. This new entity would need to be added to the running simulation. Doing this in an automated fashion is not easy, since someone needs to choose some probable actions for this new entity. OpSim handled this in the simple, barely satisfactory, manner of having the new entity move in the same direction and at the same speed at which it was last reported. If it subsequently changes direction in a significant way, this might trigger another round of agents trying to determine the impact of this change on the eventual outcome of the operation.

Once the decision-maker has been notified that the currently running simulation no longer accurately reflects the state of the actual operation, the simulation must be updated. 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.

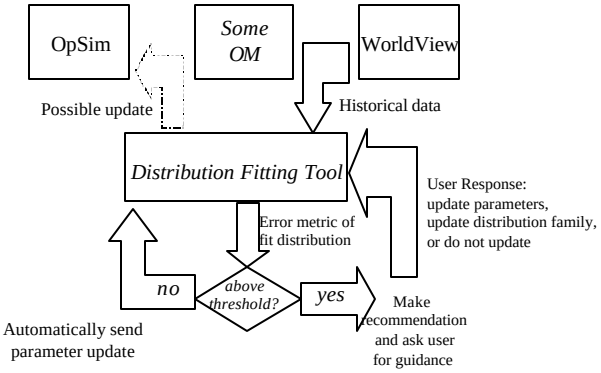

Figure 3: General Depiction of OpSim Synchronization/Updating Scheme

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, etc.) and update the future performance of the entities within the simulation. See Figure 3 for a general depiction of this process. An OM tries to refit the historical data to the family of probability distribution described for that class of entity. 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, a 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., ? and ? for a Normal distribution) are incorrect. In this case, the OM automatically sends an update message to OpSim. 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 this should make the future performance of the simulation better over time, this updating scheme makes the system adaptive.

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) an OM queries the simulation and the *stimulator* for the historical data on the various classes of entities (e.g., Bradley-equipped mechanized infantry platoons). Using the *stimulator* data as the "true" distribution, the OM conducts a Kolmogorov-Smirnov goodness-of-fit test of the data from the simulation [15, 16]. 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 correct distribution. If the new distribution is in the same distribution family as the true data, the OM tells OpSim to update the parameters of the distribution. In experiments in which the probability distribution is intentionally set to a different family of distribution, the 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 *stimulator.* Most of the changes occur immediately following small engagements, after which new data is available to the 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, OpSim does two things: it puts the entities in their current locations in the *stimulator* 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 OM uses 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 is significantly different, that 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 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. 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.

SUMMARY AND CONCLUSIONS

The use of operationally-focused simulations provides a significant enhancement to the ability of commanders to detect, evaluate, and adjust to divergences of the actual operation from the planned operation. In the OpSim prototype, Operation Monitors and other agents are combined with control systems to ensure the operational simulation remains representative of the actual operation (represented by a simulation). Also, data from the actual operation is used to fine-tune, or auto-validate, the models used in the simulation.

The anticipatory planning process accounts for the chaotic nature of warfare in which possibilities appear and disappear. The *Anticipatory Planning Support System* is designed to help commanders and their staffs see the possible flows of the battle so they can take actions early enough to influence the outcome. In testing of the anticipatory planning process, the external *Stimulator* is kept synchronized with the internal workings of the *APSS* through the use of a *Test Executive*.

The use of external simulations to stimulate planning systems like OpSim and *APSS* is a valuable testing approach. However, the implementers must pay careful attention to simulation synchronization issues. Identifying and addressing those synchronization issues will help make it possible to replace the stimulation systems with production information-provision systems once they are fielded.

REFERENCES

Gilmer, J. B., and F. J. Sullivan. 2000. "Recursive Simulation to Aid Models of Decisionmaking." In *Proceedings of the Winter Simulation Conference (WSC 2000)* (Orlando, FL, 10-13 December). IEEE, Piscataway, NY, 958-963.

Davis, W. J. 1998. "On-Line Simulation: Need and Evolving Research Requirements." Published in *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, J. Banks, ed., John Wiley and Sons, Inc., New York, 465-516.

Davis, W. J. 1998. "On-Line Simulation for Torpedo Avoidance." Available at http://www-msl.ge.uiuc.edu/~brook/boat. Last accessed on November, 1998.

Collier, K. S. 1999. "Automated Decision Support Systems Enabled by Models and Simulations -- A "Leap-ahead" Technology Recommendation for the US Army After Next Time Frame (2020-2025)." In *Proceedings of the Advanced Simulation Technologies Conference (ASTC 1999): Military, Government and Aerospace (MGA) Simulation Symposium* (San Diego, CA, April 11-15). The Society for Computer Simulation, 3-8.

Schlabach, J. L., C. C. Hayes, and D. E. Goldberg. 1998. "FOX-GA: A Genetic Algorithm for Generating and Analyzing Battlefield Courses of Action." *Evolutionary Computation*, 7(1), 45-68.

Porto, V. W., L. J. Fogel, and D. B. Fogel. 1999. "Evolving Tactics in Computer-Generated Forces." In *Proceedings of the Third Enabling Technologies for Simulation Science Conference* (Orlando, FL, 7-11 April). SPIE, 75-80.

Mitchell, T. M. 1997. *Machine Learning*, McGraw-Hill, Boston, MA.

Mason, C., and J. Moffat. 2000. "Representing the C2 Process in Simulations: Modelling the Human Decision-Maker." In *Proceedings of the Winter Simulation Conference (WSC 2000)* (Orlando, FL, 10-13 December). IEEE, Piscataway, NY 10286-1414, 940 - 949.

Davis, W. J. 1998. "Developing an Intelligent Controller for Integrated Manufacturing Resource Planning." Technical Report Draft, University of Illinois at Urbana-Champaign, Urbana, IL.

Davis, W. J. 1998. "A Framework for the Distributed Intelligent Control of Advanced Manufacturing Systems." Technical Report Draft, University of Illinois at Urbana-Champaign, Urban, IL.

Andersson, M., and G. Olsson. 1998. "A Simulation Based Decision Support Approach for Operational Capacity Planning in a Customer Order Driven Assembly Line." In *Proceedings of the Winter Simulation Conference (WSC 1998)* (Washington, DC, 13-16 December). 935-941.

Ferren, B. 1999. "Some Brief Observations on the Future of Army Simulation." *Army RD&A*, 99(3), 31-37.

Surdu, J. R., and U. W. Pooch. 1999. "Connecting the Operational Environment to Simulation." In *Proceedings of the Advanced Simulation Technology Conference (ASTC 1999): Military, Government, and Aerospace (MGA) Simulation Symposium* (San Diego, CA, 11-14 April). 94-99.

Surdu, J. R., and U. W. Pooch. 2000. "Simulation Technologies in the Mission Operational Environment." *Simulation*, 74(3), 138 - 160.

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

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

Wass de Czege, H., Jr., Personal Communication (regarding Anticipatory Planning), October, 1999.

Surdu, J. R., J. M. D. Hill, and U. W. Pooch. 2000. "Anticipatory Planning Support System." In *Proceedings of the Winter Simulation Conference (WSC 2000)* (Orlando, Florida, December 10-13). 950-957.