A DYNAMIC HIERARCHY OF RATIONAL AGENTS TO LINK SIMULATION TO THE OPERATIONAL ENVIRONMENT

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

KEYWORDS

Simulation, Operation Monitoring, Software Agents, Command and Control

ABSTRACT

This paper expands on a proposed methodology for applying simulation technology to the monitoring of ongoing operations. It focuses on the use of rational agents, called Operations Monitors, to compare the progress of the real operation with that of the planned operation. When the Operations Monitors detect differences that threaten the success of the plan, they advise the decision-maker. The purpose of such a system is to avoid information overload of decision-makers by helping focus attention on the probability of mission success. For tractability, each Operations Monitor focuses on a narrow portion of the problem domain -- there is no Napoleon agent. These Operations Monitors operate within a dynamic hierarchy that expands and contracts as necessary. This paper focuses on the structure of this hierarchy and the principle that guides the automatic and discretionary instantiation of Operations Monitors.

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] (Delany 1998). 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 (Surdu and Pooch 1998), intends to address these three technology voids. This methodology involves the use of intelligent agents, called Operations Monitors, to constantly compare the planned operation against the state of the actual operation. If the differences are significant, other tools are launched to compare these differences and make recommendations to the decision-maker.

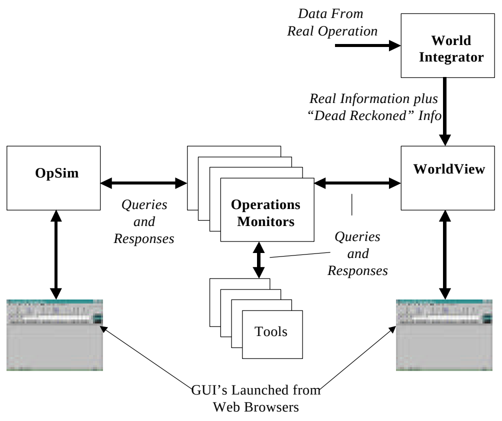

Figure 1: Proposed Methodology

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. Surdu and Pooch described the uses of simulation in support of operations. This paper concentrates on the execution phase and the manner in which the Operations Monitors

Surdu, Haines, and Pooch (Surdu and Pooch 1999) discussed the uses of simulation for COA development and analysis and for conducting rehearsals. Surdu, et. al. also described a prototype simulation in support of the overall methodology proposed by Surdu and Pooch. This paper will elaborate on the dynamic hierarchy of Operations Monitors and the principle that guides the creation of various Operations Monitors.

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

METHODOLOGY

The 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 (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 (Maes 1989; Maes 1994a; Maes 1994b). OMs should be considered part of the team, not a replacement for decision-makers (Hayes-Roth 1995). 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 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 “agent.” Franklin and Graesser (Franklin and Graesser 1997) 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.

The prototype OMs constructed to validate this methodology used fuzzy rule bases (Yen and Lengari 1999) as well as crisp rule bases. The fuzzy rule bases (built with the MATLAB Fuzzy Tool Box) are used as classifiers of information and the results of simulations. The crisp rule bases (using JESS, Java Expert System Shell, a Java implementation of CLIPS (Giarratano and Riley 1989)) are used for the control logic of the agents. 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 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 system, 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 forty-five minutes, they might launch another simulation to explore how this would effect other units. If the effect is minimal, the OM might recommend to the commander that the overall time line be shifted forty-five 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 in 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" (Gunning 1998). 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 (Army 1996). Our proposed methodology provides additional support for this process.

HOW OMS ARE CREATED DYNAMICALLY

As stated earlier, OMs focus on a narrow domain. This makes their design and implementation more 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) (Army 1993).

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 insure 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 2. 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.

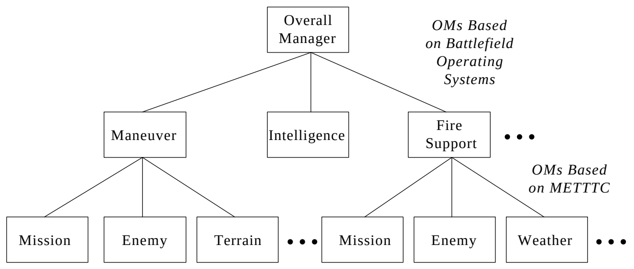

Figure 2: A Possible, Partially-Expanded Hierarchy of Operations Monitors

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 could use a variety of reasoning technologies, including crisp rule-bases, fuzzy rules bases, machine learning techniques, artificial neural networks, etc. That 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 (Army 1993). 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 the list of automatically created OMs, the mapping provides some guidance for the top-level OM. For instance in the prototype implementation, the secure mission was used in an experiment. Some analysis of a generic secure 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 secure 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. This file lists all of the subordinate OMs that the newly created OM has access to. The file indicates for which missions the subordinate OMs should be automatically launched (as described earlier). In addition, the time and space complexities are listed. It is important to note here that the absolute complexities are not estimated, but a relative comparison between the options is used. In other words the OM is only interested in which OM has a "low" time complexity, for instance. Finally this file categorizes each subordinate OM by its domain of expertise (e.g., battlefield function or METTTC).

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 the principle described earlier (based on the mission) but allow flexibility for the commander to tailor the system to the particular operation.

IMPLEMENTATION AND FUTURE WORK

A prototype OpSim that allows subscription to information and one-time queries for information by OMs has been built (Surdu et al. 1999) (see Figure 3). A small subset of Operations Monitors has also been constructed (see Figure 4). Once the simulation of the operation and the combat model

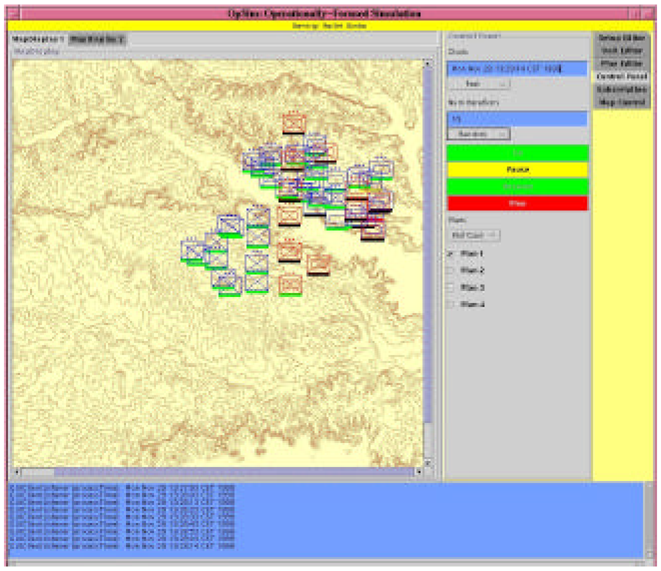

Figure 3: An Example of the Aftermath of a Brigade Attack to Seize an Objective

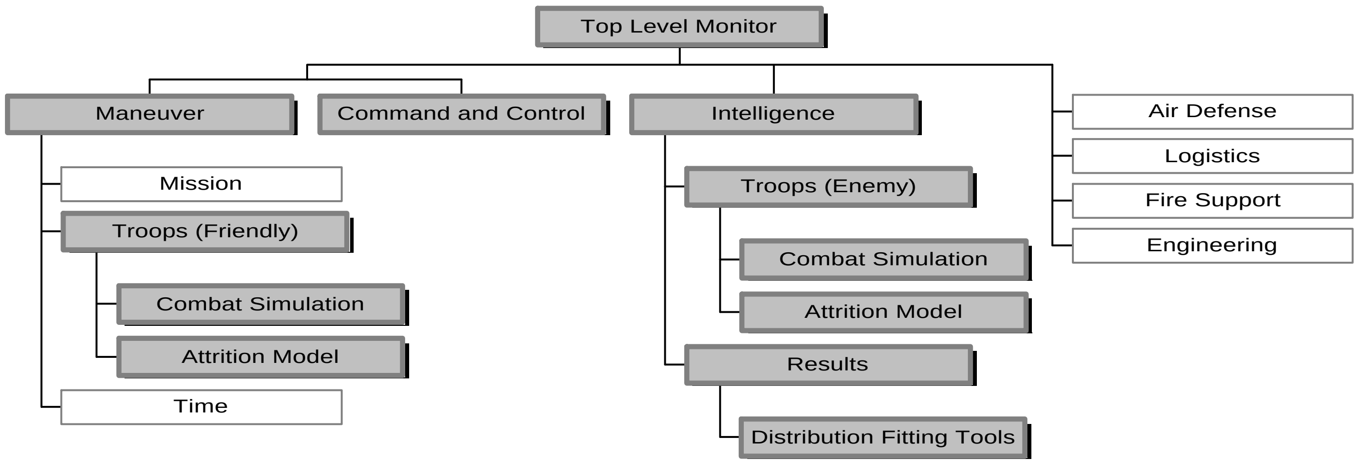

Figure 4: Implemented Operations Monitors

representing the real operation are running, the user launches the top-level OM. The top-level OM then launches a series of child monitors. The top-level OM uses the statement of the unit's mission as a guide in the initial selection of monitors to run. The user can explicitly launch and kill monitors as desired. The scenario used to test this methodology was the secure mission discussed earlier in the paper. Several scenarios were developed and four domain experts were set in front of the display of the real operation and asked to identify when the mission began to diverge from the plan and whether this deviation would have a significant impact on the outcome. In these experiments, the hierarchy of OMs was able to identify more accurately than the domain experts when deviations mattered to overall success of the mission – and did so while updating the simulation to make it a better predictor for the next mission.

As an example, one of the scenarios involved adding two, stationary enemy tank platoons on an intermediate objective of the seize mission for a brigade attack. It is not clear as the operation commences that these platoons, not present during the planning, make any difference to the attacking battalion. The human observers intuitively assumed that these tank platoons were very important and had a significant impact on the operation. In fact in most experiments the impact is at worst moderate. (Recall that this prototype uses a stochastic simulation, so the effectiveness of the entities is not the same from experiment to experiment.) The Friendly Forces OM noted the extra casualties caused by the enemy platoons, ran a simulation to explore the effects of the additional platoons, and reported moderate impact. (Figure 5 provides a screen capture of this modified scenario.)

When the previously described scenario is modified by the addition of two more tank platoons the impact on the operation becomes significant. Instead of the projected end state casualties being 5% (unmodified) or 7-9% (two additional tank platoons), the casualties are predicted at over 14%. The same result can be obtained by changing the movement of the enemy forces (to counterattack in the south, for instance) rather than adding new entities.

The rule bases used in this prototype are quite compact. The Forces Monitors, for instance (there is one which monitors each "side" in the simulation) maintain three fuzzy rule bases and one crisp rule base. The fuzzy rules were constructed using the MATLAB Fuzzy ToolBox. The prototype uses custom code to interpret and resolve these rules using the SAM method (Yen and Lengari 1999). Each of the three fuzzy rule bases has two or three inputs with a single defuzzified output and less than forty rules.

The crisp rule base handles control logic. The control of a monitor is very simple. As stated earlier, monitors can do three things: launch subordinate monitors, take advisory actions, or report results to parent monitors. The control rules were written in CLIPS code and interpreted with the Java Expert System Shell (JESS). For most monitors in the prototype, the number of rules is less than twenty.

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 have 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 *How OMs are Created Dynamically*). A variety of new scenarios need to be developed that 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

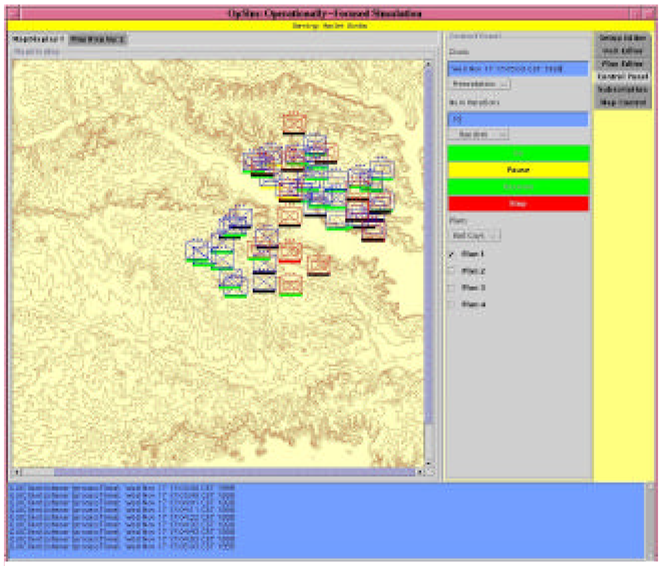

Figure 5: An Example of the Aftermath of a Modified Brigade Attack to Seize an Objective

determine what information an operationally-focused simulation needs to be able to provide.

While the bulk of this paper speaks in terms of military applications, active exploration into several civilian applications is ongoing. One particularly promising application is fighting forest fires. The planning process is very similar, and the need for exploring the effects of changing situations is very important.

SUMMARY

This paper proposed 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 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. Finally this paper described the prototype implementation that demonstrates the feasibility of the proposed methodology.

REFERENCES

Army, U. S. 1993. *FM 100-5: Operations*, Headquarters, Department of the Army, Washington, DC.

Army, U. S. 1996. *FM 100-6: Information Operations*, Headquarters, Department of the Army, Washington, DC.

Delany, P. J. 1998. “Army Modeling and Simulation Office Policy and Technology Working Group video teleconference.” personal communication.

Franklin, S., and A. Graesser. 1997. “Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents.” Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, J. P. Müller, M. J. Wooldridge, and N. R. Jennings, eds., Springer-Verslag, Berlin, 21-35.

Giarratano, J., and G. Riley. 1989. *Expert Systems: Principles and Programming*, PWS-Kent Publishing Co., Boston, MA.

Gunning, D. 1998. "Command Post of the Future." Available at http://www.mole.dc.isx.com/cpof. Last accessed on 15 October, 1998.

Hayes-Roth, B. 1995. “An Architecture for Adaptive Intelligent Systems.” *Artificial Intelligence*, 72(1), 329-365.

Maes, P. 1989. “How to Do the Right Thing.” *1180*, Massachusetts Institute of Technology, Cambridge, MA.

Maes, P. 1994a. “Agents that Reduce Work and Information Overload.” *Communications of the ACM*, 37(7), 31-41.

Maes, P. 1994b. “Situated Agents Can Have Goals.” Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, P. Maes, ed., MIT Press, Cambridge, MA, 49-70.

Surdu, J. R., G. D. Haines, and U. W. Pooch. 1999. “OpSim: a Purpose-built Distributed Simulation for the Mission Operational Environment.” In *Proc. International Conference on Web-Based Modeling and Simulation* (San Francisco, CA, 17-20 January). 69-74.

Surdu, J. R., and U. W. Pooch. 1998. “A Methodology for Applying Simulation Technologies in the Mission Operational Environment.” In *Proc. IEEE Information Technology Conference* (Syracuse, NY, 1-3 September). 45-48.

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

Yen, J., and R. Lengari. 1999. *Fuzzy Logic: Intelligence, Control and Information*, Prentice Hall, New York.