OPSIM: A PURPOSE-BUILT DISTRIBUTED SIMULATION FOR THE MISSION OPERATIONAL ENVIRONMENT

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

KEYWORDS: Discrete Simulation, Command and Control, Operations Monitor, Military, Decision Support

ABSTRACT

This paper describes a simulation built specifically for use during ongoing operations. This discrete-event simulation, implemented in Java, can be launched by a user (as an applet within a Web browser) or by other simulations. It is designed to run in near real-time so that the progress of the real operation can be compared against the planned operation. Rather than using a proprietary terrain database, OpSim reads terrain directly from VMap-2 files. This simulation also supports the capability for external agents to query the simulation for state information or ask the simulation some level of hypothetical questions while it is running.

INTRODUCTION

OpSim is a discrete event, aggregate-level, distributed simulation designed specifically for the mission operational environment. OpSim is part of an overall methodology for applying simulation technology in the mission operational environment described in (Surdu and Pooch 1998).

OpSim is unique in four main ways. First, OpSim is designed for operation across a LAN or WAN. The user interface, implemented as a Java Applet is launched from a web page and communicates with the simulation through an API. Second, there is no proprietary terrain format for OpSim as there are for many other simulations. OpSim reads data directly from VMap-2 files. Third, OpSim is designed with the ability for external agents to query the simulation while it is running. Fourth, OpSim is designed to run in a number of clock speeds and modes and to allow the user to choose what happens if the simulation begins to lag too far behind real time.

In this paper we discuss the requirements for a simulation designed for the mission operational environment and how OpSim, as part of a larger architecture, could be applied in this environment. We describe the overall architecture of OpSim. Finally we describe the various aspects of OpSim which make it unique.

REQUIREMENTS FOR A SIMULATION TO BE USED DURING OPERATIONS

The goal of the OpSim project is to build a simulation that is useful for decision-makers and their staffs during ongoing operations. In order to meet this goal, a simulation should have the following characteristics:

- The simulation must be runable 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.
- The simulation must be runable on low-cost, open systems, multi-platform environments. While much of our development effort concentrated on military applications, OpSim 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 (i.e., real time and much faster than real time). As we describe 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 OpSim to help monitor the current, ongoing operation for deviations from the plan.

- If needed, multiple simulations should be capable of operating together. While we do not see an immediate use for multiple, cooperating simulations, this simulation should be compliant with a known, accepted protocol 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.

APPLICATIONS FOR OPSIM

The purpose of OpSim is to assist decision-makers and their staffs during current operations. OpSim will be useful in both military and civilian applications. While OpSim assists staffs during the planning, rehearsal, and execution phases of an operation, it is designed specifically to facilitate decision making during execution. In this design process, we have focused on military ground combat operations and fighting forest fires.

While the various 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 could be a useful tool in each of these phases.

During the planning phase, staffs develop courses of action (COAs). This involves proposing a number of possible means of accomplishing the mission. At the same time, the staff identifies the most likely and most dangerous enemy COAs. With a purpose-built simulation, the staff enters the enemy and friendly courses of action and simulates them. After several iterations, the simulation tool rates the various COAs using specified decision criteria. These ratings are an additional source of information the commander uses in choosing which COA to pursue.

Once a COA has been chosen, it is then developed into a full plan. The simulation might also facilitate this process. Once the plan is finalized, it can be played back for the commander, his staff, and the subordinate leaders as part of a rehearsal. At critical moments the simulation can be halted and the conduct of the operation at that point can be discussed. The simulation can also be rolled back or fast-forwarded to the previous or next critical event, respectively.

Once the plan has been chosen, refined, and rehearsed, and the operation commences, OpSim is constructed so that external agents can query the simulation for information and compare the progress of the simulated plan and the real operation. When significant deviations from the plan occur, these external agents can launch tools that explore the impact of these deviations. Finally the commander is advised of significant changes.

As an example, while a forest fire fighting operation is commencing, the wind may shift directions from that forecasted in the plan. The external agents proposed by Surdu and Pooch (Surdu and Pooch 1998) can compare the predicted path and spread of the flame with that in the plan. This comparison may indicate that additional manpower is needed, that different areas need to be evacuated, etc. In a military operation, the sudden appearance of a large enemy body in the real operation versus the plan may cause an external agent to launch another simulation to determine the probable impact of this new unit. In each case, the fact that OpSim's design allows external agents to monitor its state facilitates this comparison of the *real* operation with the *planned* operation†.

DEFINITIONS USED IN THIS PAPER

Aggregate-Level Simulation – a simulation in which the entities represent groups of people and machines rather than individual people and machines.

Stochastic Processes – processes that contain a certain amount of randomness in their transitions from one state to another. These systems are nondeterministic in that the next state can not be unequivocally predicted based on the present state and stimulus (Pooch and Wall 1993).

Near Real Time -- data and displays representing the real-world with a specified latency.

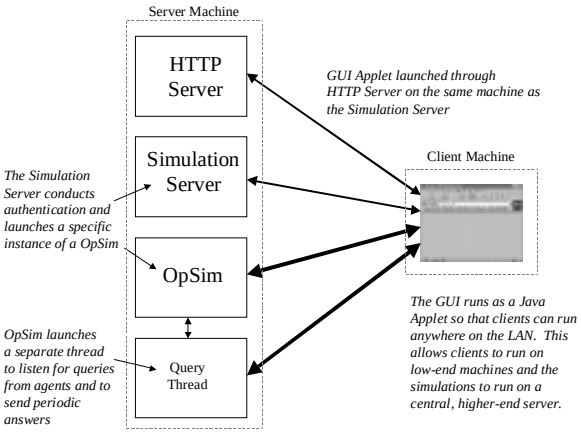
ARCHITECTURE OF OPSIM

The Defense Advanced Projects Research Agency (DARPA) has recognized the importance of simulation in command and control activities. In its concept briefing for the Command Post of the Future

† For a more detailed discussion of the interaction of OpSim with agents and the real world, see Surdu and Pooch 1998.

(CPoF) project, DARPA lists 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 (DARPA 1998).

The overall architecture of OpSim is designed around the requirements stated earlier. While the simulation itself runs on a server machine (which may have significant computational power if required), the interface to the simulation is a Java Applet which can run on relatively low-end machines. As part of the Java security management scheme, Java Applets can only connect to machines from which the applet code resides. For this reason, a small HTTP Server is required to run on the same machine as the Simulation Server.



The GUI applet makes a TCP/IP connection to the Simulation Server. After authentication and the passing of certain exercise parameters (e.g., port number and exercise identification), the Simulation Server launches a specific instance of OpSim and hands off the TCP/IP connection of the GUI to the newly-created OpSim process. When other GUI's connect to the Simulation Server, other simulations are launched. If these simulations must collaborate, they do this through the Aggregate Level Simulation Protocol (ALSP) like protocol, as discussed later.

OpSim *is* the simulation. The movement of entities, the resolution of interactions (e.g., attacks, effects of fire on trees, etc.), and the management of time are resolved by OpSim. Control over the simulation is exercised by the user, through the GUI, which communicates to OpSim through an API. For instance, the user tells OpSim which plan file to load through the GUI. This plan file is critical because it contains the list of entities and their planned actions. Through the GUI, the user also specifies the terrain database.

The terrain database physically resides both at the Server and Client machine. This is due to the large size of the database and its static nature. The plan file, however, resides only on the Client machine. This eliminates the need for the Server machine to manage the plan files of several clients. Maintaining the plan files locally requires the GUI applet to read a plan file from the local file system and transmit the information to OpSim. Normally Java Applets are not allowed to access the local file system. In order for the GUI applet to read the plan file, the GUI applet is a signed Java Applet (Sun Microsystems 1998).

OpSim launches some number of additional threads that manage the receipt and periodic answering of queries from connected agents. OpSim treats the GUI just as any external agent which requests information (e.g., state of the simulation's entities). The GUI connects to the Query Thread as part of the initialization process. Other agents that wish to query OpSim for information must connect to OpSim through it's published port number. OpSim then launches another Query Thread to service the connecting agent. Clearly there is a maximum number of these threads that can be launched before the simulation(s) begins to lag behind real time. This maximum number is largely determined by the size and number of CPU's on the server machine and the amount of virtual memory available to running processes.

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 postings 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, we felt that the process of converting digital terrain information to some proprietary format was burdensome. For the sake of performance and usability, we searched for a better representation of terrain data for use in OpSim. We found a solution at the National Imagery and Mapping Agency (NIMA) (NIMA 1998a).

OpSim uses terrain data in Vector Product Format (VPF™)*. VPF is a standard format, structure, and organization for large geographic databases and is

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

based on a relational data model. VPF allows applications to read terrain data directly from computer-readable media without converting it to an intermediate format. Terrain data in VPF is in vector format that concerns itself with vertices, lines, and xyz coordinates. This allows for data to be accessed by spatial location and thematic content (an advantage over the raster model for terrain data) (DoD 1996a, NIMA 1998b).

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 (DoD 1996b). For geographic information that may not be included in a VMap-2 library, OpSim will maintain a "delta-file" of objects, for each library, that need to be displayed as part of the terrain.

To display the correct terrain data, when a user requests the start of a particular simulation, the Simulation Server will look for the VMap-2 library, or sub-area within the library, associated with the plan file and pass that information back to the GUI applet. The hosts on which the Server and the applet execute each have their own local copies of the data libraries so that terrain data does not have to be passed across the network.

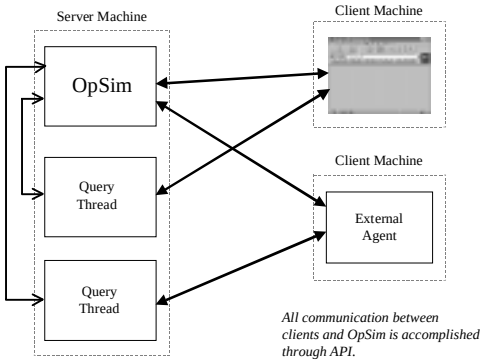
QUERY CAPABILITY AND API

At this time, OpSim only supports pre-defined queries implemented through a series of calls in the API. In the future, we intend to make OpSim capable of answering ad hoc queries through some query language such as KQML (Bradshwa 1997).

All communication with the Simulation Server and an OpSim is conducted through a set of API calls that we developed. The Server and a running simulation are stand-alone applications with no intrinsic or organic interface except for the API. Whether it is an external agent or a GUI applet sending messages, any connected client uses the same common API.

The API calls provide for a wide range of functionality. Calls exist to:

- Initiate and start the execution of a plan.
- Manipulate the execution clock.
- Add/update/delete entities.
- Query entities.
- Register interest in receiving periodic updates on a particular group of entities.
- Request server health/status information.
- Establish levels of secure communication.



When the Server receives the calls, it processes them on a FIFO basis. No priority levels exist among the different calls. Not every external client will necessarily have access to the full set of API calls. For instance, the external agent has no need to be able to manipulate the execution clock of the simulation, while the GUI applet might. Also, care is taken when two or more external clients want to update or change the state of the same entity.

NEAR REAL TIME CONSTRAINTS

For the simulation to be of use to the operational decision makers, it must produce timely information with minimal latency between what it displays as the state of a particular object and the real world state of that object.

To maintain an accurate state of simulation entities while at the same time trying not to impact system performance, we established a general update latency constraint of 300ms for all simulation entities. A general guideline within the real-time simulation community is that humans can't detect differences in time less than 100ms (DIS Steering Committee 1994). We have relaxed that guideline since OpSim works at the aggregate level. This constraint applies to all update instances whether the update is initiated by the user, an external agent, or by the simulation process itself. In times of moderate simulation activity, this constraint holds. As the activity level increases (i.e., a battle engagement or the wind shifts during a forest fire) this constraint can cause the simulation to lag far behind the real world.

In situations of high-activity level, OpSim allows the user, through an API call, to set how the update of entity states should be handled. At times, the user may want all of the simulation entities to play and may not be concerned about how far behind the simulation becomes compared to real world time. At

other times, the user may be extremely concerned about latency. In this case, a priority level must be given to each entity for state updates. The state of higher priority entities will be at most 300ms behind the state of the real-world object that it represents while some of the lower priority entities may be anywhere from a few seconds or minutes behind to not being updated at all. The lower priority entities will "catch-up" when the activity level reduces.

ATTRITION/INTERACTION MODELING

The purpose of this project is not to develop a valid attrition model. It *is* our intent to include an attrition module and allow for this object to be easily replaced by some other attrition model if another is desired. In looking at attrition models, we compared models from the Military Operations Research Society (MORS) (Military OR Society 1995), models based on Lanchester equations (Davis 1998), and the model proposed by Dupuy(Dupuy 1995). All of these models have known limitations. For instance, Lanchester equations are only valid for frontal engagements by nearly equally competent opponents (Davis 1998). Finally we decided to incorporate the Dupuy model, because it is easy to implement and computationally inexpensive. We emphasize that some other model can easily replace this attrition model without affecting other subsystems. As proof of this notion, we also included a simple interaction model for fire fighting as well as the Dupuy attrition model.

Our implementation of the Dupuy model required slight modification. First, the Dupuy model was designed for large units (e.g., divisions). We extrapolated one of the many charts to allow for predictions of smaller bodies (e.g., platoons). The Dupuy model computes daily casualty rates. In OpSim, we are interested in engagements that last an hour or so. In (Dupuy 1995) Colonel Dupuy allows that practitioners of his model would need to compute the number of casualties for a smaller period of time, and we chose to do that. Finally, the Dupuy model, like the closed form of Lanchester Equations, is deterministic. We added a term to the Dupuy casualty rate which follows a standard normal distribution (mean of zero and standard deviation of one). Again, the focus of this research is not the development of an attrition model, but Dupuy's model, as modified, gives results which exhibit good face validity (DARPA 1998).

COMMUNICATION

As discussed earlier, one of the goals of this simulation is that it eventually be able to interact with other simulations in a distributed manner. Since OpSim is an aggregate-level simulation, a reasonable communications protocol is one like the Department of Defense (DoD) Aggregate Level Simulation Protocol (ALSP). ALSP grew out of a DARPA research effort to create a message-based protocol for linking distributed, constructive simulations. ALSP also includes a time and data management infrastructure and a architecture which assists the development of new, ALSP-compliant simulations.

Current ALSP-compliant simulations include the Army Corps Battle Simulation (CBS), the Air Force Air War Simulation (AWSIM), the USMC Marine Air Ground Task Force (MAGTF) Tactical Warfare Simulation (MTWS), and many others. The ALSP confederation is the largest confederation of simulations (ALSP 1998a, ALSP 1998b, MITRE 1993). As future work, we propose to make OpSim ALSP compliant, because ALSP is a known, proven protocol, and ALSP was designed to support aggregate-level simulations. Finally, future versions of the ALSP Infrastructure Software (AIS) will be High-Level Architecture (HLA) compliant (DMSO 1998).

SIMULATION CLOCK CONTROL

Each copy of OpSim is presumed to run in real, or wall-clock, time. The user usually determines the start time and the starting point within the simulation, and control of the simulation clock can be accomplished through the API. There may be moments when the simulation must run faster than or in multiples of wall-clock and when it must process at high-speed.

If OpSim is being used for planning or exercise purposes, the user has the option of moving forward or backward to particular points in time within a simulation. To move forward in time, OpSim must process the simulation as fast as possible in order to place the simulation into the environmental state at the requested point in time. The user may also wish to run an entire simulation within a shortened time period, thus asking OpSim to execute in multiples of wall-clock time, i.e. 2x or 6x faster.

When being used for operational purposes, the simulation will most likely run at wall-clock time, but if it is asked to run a "what-if" scenario by an external agent, the "what-if" scenario will be processed at high-speed by a separate thread so as to provide an expeditious answer.

PLAN REPRESENTATION

In OpSim, the aggregate entities follow a predefined plan. A plan consists of a series of

connected nodes. These nodes contain actions and preconditions. These nodes may also contain decision points and branches. This allows the nodes of a given entity to have multiple input connections and multiple output connections.

This structure was devised to facilitate inferences by external agents. Rather than waiting for some action to fail, an external entity can attempt to determine the likelihood that all of an actions preconditions will be met. If this external agent looks sufficiently far ahead of the current state of the simulation, the agent many be able to give advanced warning to decision makers in time for them to adjust the plan accordingly.

SUMMARY

OpSim is unique in four ways. It was designed for use in a web-based environment by making the user interface a Java Applet, which is launched from a web page. OpSim does not have a proprietary terrain format, but rather it reads the VPF databases provided by NIMA directly. OpSim was designed to allow external agents to query the simulation through the API. Finally OpSim allows the user a large degree of control over the simulation clock and the control of the near real time constraints.

OpSim is part of an overall architecture designed to apply simulation technologies to the mission operational environment. The unique aspects of OpSim and its compliance with the requirements stated in (Surdu and Pooch 1998) make it ideally suited for that environment.

REFERENCES

ALSP 1998a. Aggregate Level Simulation Protocol. URL: http://alsp.ie.org/alsp.

ALSP 1998b. ALSP 89-92 History. URL: http://ms.ie.org/alsp/89-92_history/89-92_history.html.

Bradshwa, J.M. 1997. "KaoS: Toward An Industry Strength Open Agents Architecture," In *Software Agents* (DRAFT), J.M. Bradshwa, ed. AAAI Press.

DARPA 1998. Defense Advanced Research Projects Agency Command Post of the Future web site. URL: http://mole.dc.isx.com/cpof and http://www-code44.nosc.mil/cpof.

Davis, P.K. 1998. "Aggragation, Disaggregation and the 3:1 Rule in Ground Combat." Rand publication at URL: http://www.rand.org/publications/MR/MR638.

DIS Steering Committee. 1994. *The DIS Vision: A Map to the Future of Distributed Simulations.* Institute for Simulation and Training, Orlando, FL.

DMSO 1998. Defense Modeling and Simulation Office HLA Home Page. URL: http://hla.dmso.mil.

DoD 1996a. MIL-STD-2407, Department of Defense Interface Standard for Vector Product Format, Fairfax, Virginia: NIMA, 28 June 96.

DoD 1996b. MIL-V-89032, Military Specification: Vector Smart Map (VMap) Level 2 (Draft), Fairfax, Virginia: NIMA, 15 April 1996.

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

Military OR Society. 1995. *Military OR Analyst's Handbook Volume II: Conventional Weapons Effects.* S.H. Parry, ed. Military Operations Research Society, Alexandria, VA.

MITRE 1993. “Aggregate Level Simulation Protocol (ALSP) Program Status and History”. MITRE Corporation, 7525 Colshire Drive, McLean, VA, 22102 (Mar.).

NIMA 1998a. National Imagery and Mapping Agency Public Web page. URL: http://www.nima.mil.

NIMA 1998b. VPF Overview Web page. URL: http://www.nima.mil/vpfproto/index.htm.

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

Sun Microsystems 1998. Signed Applts, Browsers, and File Access Web page. URL: http://developer/java.sun.com/developer/technicalArticles/monicap/Security/Tools/signed.html.

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