KNOWLEDGE-BASED SIMULATION USING OBJECT-ORIENTED PROGRAMMING

Karen M. Sidoran

Air Force Materiel Command
Rome Laboratory
525 Brooks Rd.
Griffiss AFB, NY
13441-5700

ABSTRACT

Simulations have become a powerful mechanism for understanding and modeling complex phenomena. Their results have had substantial impact on a broad range of decisions in the military, government, and industry. Because of this, new techniques are continually being explored and developed to make them even more useful, understandable, extendable, and efficient. One such area of research is the application of the knowledge-based methods of artificial intelligence (AI) to the computer simulation field.

The goal of knowledge-based simulation is to facilitate building simulations of greatly increased power and comprehensibility by making use of deeper knowledge about the behavior of the simulated world. One technique for representing and manipulating knowledge that has been enhanced by the AI community is object-oriented programming. Using this technique, the entities of a discrete-event simulation can be viewed as objects in an object-oriented formulation. Knowledge can be factual (i.e., attributes of an entity) or behavioral (i.e., how the entity is to behave in certain circumstances).

Rome Laboratory's Advanced Simulation Environment (RASE) has been developed as a research vehicle to provide an enhanced simulation development environment for building more intelligent, interactive, flexible, and realistic simulations. This capability will support current and future battle management research and provide a test of the object-oriented paradigm for use in large scale military applications.

INTRODUCTION

The adoption and incorporation of the various knowledge-based methods of artificial intelligence (AI) into the simulation process provides benefits in a number of areas. Some of these areas include: knowledge representation in the simulation model, decision making within a simulation, rapid prototyping of models, data analysis of simulator-generated outputs, and model modification and maintenance (Nielsen, 1991).

The problem of representation is central to all of AI. Eight paradigms that have found favor in today's practice are [Garcia-91]:

- Semantic networks
- Frames and scripts
- Procedural representations
- Analogical or direct representations
- Specialized languages for knowledge representation
- Object-oriented programming
- Logic representations
- Rule-based representations

This paper will focus mainly on the technique for representing and manipulating knowledge known as object-oriented programming. In many practical situations, the eight representation paradigms previously mentioned are combined in hybrid systems. An example of this integration will be discussed in a latter section of this paper which discusses an extension of our object-oriented research to incorporate a rule-based approach.

OBJECT-ORIENTED PROGRAMMING

The object-oriented approach attempts to manage the complexity that is characteristic of real-world problems by abstracting out information and encapsulating it within objects [Wirfs-Brock-90]. Using this technique, the entities of a discrete-event simulation can be considered as objects in an object-oriented formulation. There are two types of knowledge associated with each object, factual and behavioral. An object's factual data can be viewed as the attributes of the entity, characterizing its capabilities, current state, and parameters. The object's behavioral knowledge characterizes how the entity is to behave in certain circumstances. Messages sent to an object can provide factual data (such as recalling an attribute) or initiate a specific behavior (such as asking an air mission to fly its route).

This type of representation and processing supports rapid prototyping in that particular model functions can be changed or replaced with minimal impact on other sections of the model. It also facilitates the modification and maintenance of models. Often we can organize objects into a family of homogeneous classes, which are mutually distinct, but do share certain fundamental properties in common. Within the object-oriented paradigm, a class definition can specify from which classes a new family of objects will inherit (i.e., automatically obtain factual and behavioral knowledge). This saves having to copy or rewrite common variables and methods and helps maintain consistency when code modifications and model extensions are made [Zeigler-90].

ROME LABORATORY'S ADVANCED SIMULATION ENVIRONMENT (RASE)

Since 1986, Rome Laboratory (RL) has been exploring ways to utilize the advantages offered by the object-oriented paradigm to improve Air Force battle management simulation [Anken-91]. The focus of RL research has centered on building an environment that will allow us to develop and test object-oriented simulations.

The Rome Laboratory Advanced Simulation Environment (RASE) began with the development of an object-oriented language known as Enhanced ROSS [McArthur-82] in Common Lisp (ERIC)[Hilton-90]. To exploit the capabilities of ERIC, the program was expanded to include the development of an object-oriented Cartographic system (CARTO), a Map Display System which serves as the graphical interface to the CARTO system, and an interactive object-oriented battlefield simulation known as the Land Air Combat in ERIC (LACE) simulation. Other support tools were also developed and include an object hierarchy browser (HIER), an object editor and run time instance viewer (RACK), and a simulation clock manipulation tool (Clock Viewer) [Anken-91]. Currently, RASE is geared toward the tactical battlefield domain and includes a cartographic system which covers a portion of Central Europe. However, many of the RASE tools are generic and can be used to develop simulations in other domains.

LACE, which was built as a part of, and with the tools of the RASE environment, simulates the tactical engagement of NATO and Warsaw Pact forces in Central Europe. The simulation's primary focus is on the air side of the battlefield model. While the main purpose of LACE has been to evaluate the object-oriented paradigm in a large-scale military application, it has also been designed to provide a realistic environment for evaluating tactical decision aids and to support tactical commanders in their decision making process.

LACE currently includes objects for:

- air-facilities
- wings & squadrons
- fighter, cargo, & refueling aircraft
- surface-to-air missile (SAM) sites
- runways
- petroleum-oil-lubricant facilities (POL)
- munition targets

Missions include offensive counter-air, transport, refueling, and SAM Suppression. The overall system includes hundreds of stationary and mobile objects able to interact in the simulation. The current demonstration includes six air missions with 32 SAM units and radars set up to defend enemy targets.

With continual map display updates the simulation runs approximately 38 times faster than real-time. Running LACE without graphics has resulted in speed-ups of over 70 times real-time [Anken-91].

CURRENT RESEARCH ISSUES

Through our work with RASE we have found that object-oriented programming lends itself very well to battlefield simulation. The modern-day battlefield is a complex domain with thousands of autonomous agents. Each agent, or object, has its own goals and behaviors while functioning within the goals and plans of higher entities. Object-oriented programming provides a natural means of representing these entities.

We feel that our research efforts have effectively addressed many of the problems with traditional military simulations. Most of these simulation environments lack features essential to adequately support Air Force needs. They lack flexibility, are difficult to embed knowledge into, are expensive to build and maintain, and generally are not compatible with current knowledge-based decision aids [Anken-89]. In contrast, the RASE environment provides a highly interactive, intelligent, and flexible simulation capability.

Through our development of RASE we have identified other areas of potential research for improving simulation techniques. One such area is improving the inspectability of our simulation models. Without the capability to thoroughly inspect and understand the underlying model, it is difficult to validate (ensure that the right system was built) and verify (ensure that the system was built right) the model that is being or has been developed.

Another potential area of improvement is execution speedup through distributed processing. If simulations could be made to run more rapidly, users could experiment with more battlefield strategies in a shorter length of time. These issues will be briefly discussed in the following sections.

COMBINING RULE AND OBJECT-ORIENTED APPROACHES

While the object-oriented approach provides inspectability in terms of the attributes of the entities being modeled and how entities are related within the model, there is no information explicitly available in terms of the model behavior (i.e., the when, where, why, and how's of the events associated with object actions). Thus, the logic of why an event occurs, or does not occur, is lost within the body of program code. Inspecting simulation models, such as LACE, usually requires the user to review the source code to fully understand the interactions between simulated entities. Understanding the results of the simulation is often accomplished by saving (usually to disk or hard copy) selected data items as the simulation progresses.

This lack of explicit knowledge severely limits the types of questions traditional state-based simulations can answer. These limitations are most evident in the area of model inspectability where the user would require the simulation to explain its behavior in meaningful terms.

Understanding the results of the simulation model includes statistical and causal information [Grimshaw-92]. Statistical information includes both the history of the simulation and derived historical information. History would include both events and temporal state information. For example, in the LACE model, the user may want to know what happened to a given air mission on its way to the target or the location of that air mission at a certain time. Derived historical information might include averages or standard deviations of events and temporal states. The user may want to inquire about the average expected target damage or the chances of a mission being successful.

Causality refers to the events or states which lead to the occurrence of a specific event. One may want to ask why a certain air mission was shot down or why another mission was successful. Using the object-oriented approach of RASE, this information was not readily available. Queries were limited to "What if" questions, allowing the user to interactively stop the simulation, make changes to object attributes, create or destroy objects, and restart the simulation to see what happened.

To overcome these limitations, the combined use of a rule-based approach with an object-oriented approach was proposed [Grimshaw-92]. In this hybrid approach, time-lines are used to capture temporal values and causal events, providing history and causality to the simulation. The modeling language that was developed

at Rome Laboratory to demonstrate this concept is an extension to ERIC known as DeclERIC (Declarative ERIC). The work in this area incorporates the advantages of both rule-based and object-oriented techniques, providing the ease of representation with the explicit representation of behavioral knowledge.

Time-lines are created that capture what an object's selected attribute values will be for the length of the simulation assuming there are no interactions between objects. For example, if a SAM site has 5 missiles initially, we assume it will have 5 missiles forever. If an aircraft is given a route, we assume it will fly this route without any change in course or speed. Then, leveraging off TMM (Time Map Manager, a system developed by Tom Dean) [Dean-85; Dean-87], the programmer defines Spatial-Temporal Possibilities (STP's) of interest [Grimshaw-92], such as when a mission will be over its target, or when a mission is going to be in-range of a SAM site.

Using the STP's and the time-lines as pre-conditions, DeclERIC will create all the possible bound rules from the user-defined rule base. A rule example is the SAM-fire-rule which states that if a mission Y is in-range of SAM X, it's jammer is off, and SAM X has more than one missile, then decrement the number of missiles of SAM X and update the status of mission Y if it has been damaged.

After binding all the rules it can, DeclERIC picks the earliest one, discarding the rest, and executes it, changing whatever object attributes that need to be changed according to the postconditions of the executed rule. Then it cycles through, sending TMM the new time-lines, receiving a new set of STP's, and finding the next rule to fire. DeclERIC is finished when there are no more rules that can fire. After the simulation has executed, the user can watch it play on a graphical display and ask menu-driven questions about what happened and why. The value of object attributes now reflect their full history and a list of rules that have been fired is maintained by DeclERIC. Work in the area of declarative simulation and DeclERIC is still underway.

DISTRIBUTING AN OBJECT-ORIENTED SIMULATION

A second issue that was previously mentioned is that of simulation execution speed. Although the object-oriented paradigm made LACE easier to change and analyze than previous simulations, it was discovered that execution time could still present a problem when thousands of objects must interact in a very large simulation. This opened another avenue of research which has been to study the feasibility of extending an object-oriented simulation into a distributed paradigm. During this project, DERIC, a distributed version of ERIC, was developed at Rome Laboratory [Lawton-92].

The development of DERIC has shown that this extension into the distributed/parallel paradigm is feasible [Lawton-91]. As a result of this research, several properties of ERIC were identified that must be preserved in a distributed extension, such as causality and user interactivity. Also, a number of critical issues inherent to parallel simulations, including deadlock avoidance, atomicity, and event monotonicity were encountered [Lawton-92]. While it has been concluded that distributed computing extensions to ERIC are desirable, speedups of distributed object-oriented simulation languages are not easily realized [Lawton-91]. Research in the area of distributed object-oriented simulation is still actively in progress.

CONCLUSION

Knowledge-based simulation has the potential to provide a powerful mechanism for understanding and modelling complex phenomena. The goal of this area of research is to build simulations that are more powerful and more comprehensible by making use of deeper knowledge about the behavior of the simulated world. Such knowledge is usually omitted from traditional simulations since they are unable to utilize it. This lack of explicit knowledge severely limits simulations and makes it difficult to verify the correctness of their model, makes them difficult to comprehend, and restricts the kinds of questions they can answer. A general lack of formal validation techniques has limited the use of knowledge-based simulation in critical applications. This has contributed to the fact that the number of operational systems is significantly less than the number of systems developed. However, the need for "intelligent" simulations is evident and knowledge-based paradigms such as object-oriented programming and rule-based inferencing provide the mechanism for advancing the development of computer simulation models in many domains.

REFERENCES

[Anken-89] Anken, Craig S., "LACE: Land Air Combat in ERIC," Rome Lab In-House report RADC-TR-89-219, Rome Laboratory, Griffiss AFB, New York, October 1989.

[Anken-91] Anken, Craig S., "An Air/Land Combat Simulation and Scenario Generation Assistant," PROCEEDINGS OF THE 1991 INTERNATIONAL SIMULATION TECHNOLOGY CONFERENCE, Orlando, Florida, October 21-23, 1991.

[Dean-85] Dean, T., "Temporal Imagery: An Approach to Reasoning about Time for Planning and Problem Solving", Tech. Report 433, Computer Science Department, Yale University, New Haven, Connecticut, 1985.

[Dean-87] Dean, Thomas L. and McDermott, Drew V., "Temporal Data Base Management", AI Journal, Elsevler Science Publishers, 1987.

[Garcia-91] Garcia, Oscar N., and Chien, Yi-Tzuu, KNOWLEDGE-BASED SYSTEMS: FUNDAMENTALS AND TOOLS, IEEE Computer Society Press, Los Alamitos, California, 1991.

[Grimshaw-92] Grimshaw, Capt Jeffrey, "Simulation Modeling Using an Integrated Rule and Object-Oriented Approach," Unpublished Rome Lab In-house Report, Rome Laboratory, Griffiss AFB, New York, January 1992.

[Hilton-90] Hilton, M., and Grimshaw, J., "ERIC Manual," Rome Lab In-House Report RADC-TR-90-84, Rome Laboratory, Griffiss AFB, New York, April 1990.

[Lawton-91] Lawton, James H., Krumvieda, Clifford D., "Distributing Object-Oriented Simulation Languages," PROCEEDINGS OF THE 1991 SUMMER COMPUTER SIMULATION CONFERENCE, Baltimore, Maryland, July 22-24, 1991.

[Lawton-92] Lawton, James H., Krumvieda, Clifford D., "DERIC: A Distributed Object-Oriented Simulation Language," NATO WORKSHOP ON OBJECT-ORIENTED MODELING OF DISTRIBUTED SYSTEMS, DREV, Quebec, Canada, 1992.

[McArthur-82] McArthur, David and Klahr, Philip, "The ROSS Language Manual," N-1854-AF, The Rand Corporation, Santa Monica, 1982.

[Nielson-91] Nielson, Norman R., "Application of Artificial Intelligence Techniques to Simulation," KNOWLEDGE-BASED SIMULATION METHODOLOGY AND APPLICATION, Vol. 4, Springer-Verlag New York Inc., New York, New York, 1991.

[Wirfs-Brock-90] Wirfs-Brock, Rebecca, et. al, DESIGNING OBJECT-ORIENTED SOFTWARE, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[Zeigler-90] Zeigler, Bernard P., OBJECT-ORIENTED SIMULATION WITH HIERARCHICAL, MODULAR MODELS, Academic Press Inc., San Diego, California, 1990.