Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

JOINING A DISTRIBUTED SIMULATION ENVIRONMENT VIA ALSP

Lydia P. Dubon

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109-8099

ABSTRACT

The Aggregate-Level Simulation Protocol (ALSP) provides a common platform where a simulation can import objects owned by other simulations (ghosting) into its local modeling framework, and can export its objects to other simulations. In ALSP, simulations are referred to as actors, and an aggregate of simulations form the ALSP Confederation. The first pair of actors to successfully join the confederation are the Corps Battle Simulation (CBS) and the Air Warfare Simulation (AWSIM). CBS is currently undergoing changes to interface with the Combat Service Support Training Simulation System (CSSTSS).

This paper describes the approach and changes that allow CBS to successfully join the ALSP Confederation. There are two distinct areas of changes that CBS had to undertake: functional changes and interface changes. The functional changes give CBS the ability to generate the simulation event data and object state changes needed to support the modeling interface with other actors. This was achieved by enhancing or inhibiting certain functionality in the model. The interface changes provide CBS with the ability to recognize and generate ALSP messages needed to support the functional changes. The implementation approach to the above changes was based on the philosophy that CBS's stand-alone functionality should be kept intact.

1 INTRODUCTION

The purpose of this paper is to summarize the software experience and implications of joining the ALSP Confederation from an actor's point of view, the actor being CBS. The emphasis is on the interface changes recently applied to CBS 1.4, which enable it to function in the distributed simulation environment provided by ALSP.

The ALSP documentation refers to the actor's TRANSLATOR as the segregated piece of code containing most of the modifications to the actor. The CBS translator does contain all the interface changes. It should be noted, however, that depending on the functional interface agreed to with the other actors, significant code changes may take place deep within the modeling portion of the simulation. This is the case with the CBS-AWSIM and CBS-CSSTSS interfaces. For example, Tactical Air functionality is now switchable code when AWSIM is part of the confederation. The CBS translator does not produce modeling data; it only relays it in the form of ALSP messages.

The following sections concentrate on the implementation of the CBS translator and on the changes inside CBS needed to support it. Section 2 briefly describes the CBS software. Section 3 provides a brief overview of the modeling changes. Section 4 presents an overview of the CBS translator. Section 5 describes the ALSP parser and verifier implementation. Sections 6 and 7 present CBS-ALSP time and data synchronization, respectively. Section 8 describes the data mapping purpose of the translator. Finally, Section 9 discusses the implications of interfacing with additional actors.

2 CBS SOFTWARE OVERVIEW

The implementation of the CBS translator took advantage of CBS 1.4 features. Therefore, a brief overview of CBS is presented here. CBS is a discrete event simulation that provides computer-based battle simulation support for Command Post Exercises. These exercises are used to train command and staff officers at the joint task force, corps, division, and brigade levels. The basic components of the CBS software system are the combat model (CBS executable); the workstation-graphics, and communications programs; and the database software. The combat model is written in the SIMSCRIPT II.5 language, and the workstation and translator software are written in C. Part of the CBS executable is the Executive, which provides a powerful window into the model. The Executive module consists

of a FORTH interpreter, terminal and mailbox I/O drivers, and the timing routine. The running of the CBS model relies on a library of FORTH routines used by the Executive. The CBS translator communicates directly with the CBS Executive.

The CBS 1.4 system includes a library of Master Interface (MI) utilities that provides two-way communication between CBS and an external application. The MI utilities provide access to a workstation database that is exclusive to the application and resides on the same node. It is referred to as the MI database. This database is a repository of CBS objects and associated data. As an MI application, the CBS translator uses the MI database to obtain simulation objects' state data during normal processing and crash recovery. The database is a reflection of the latest CBS object changes that occurred in the CBS model.

3 MODELING CHANGES

The CBS combat model can be subdivided into six top-level functional areas: Maneuver, Artillery, Chemical Warfare, Combat Engineer, Logistics, and Air Operations. When CBS and AWSIM are joined to the confederation, CBS must ghost AWSIM tactical air missions and provide the confederation with units owning air defense weapon systems. This interface mostly involves inhibiting certain functionality in the area of Air Operations. In the CBS-CSSTSS interface, which is currently under development, CBS will create and move convoys for CSSTSS, and register all ground combat units. CSSTSS will register ground support units and control equipment and personnel for all units. This interface involves many functional changes in the logistics area of the model. In both of these interfaces, the modelers had to make changes to produce the simulation events and data expected by the other confederation members. It is not a trivial task to modify existing functionality in the model.

4 CBS TRANSLATOR OVERVIEW

The CBS 1.4 ALSP translator is referred to as the ALSP MI Application (ALMA). It is an independent process that runs on a dedicated computer along with the database software. Aside from the advantages of being an MI application, it was a requirement to make the translator a separate process to avoid taking away from the CBS model any CPU utilization needed for ALSP message processing. The desire is not to burden CBS with model-independent ALSP processing and not to affect stand-alone functionality. The ALMA uses CBS orders and control messages to drive confederation requests into CBS. The orders are similar to those a user would send from the menu at the CBS workstation. The ALMA expects CBS simulation events and object state changes to be communicated via CBS event messages and game truth, respectively. Figure 1 is a data flow diagram of the CBS translator.

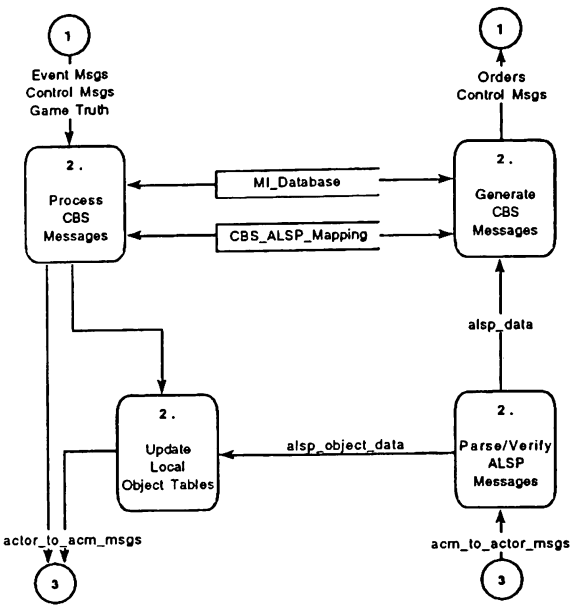

Figure 1: ALMA Data Flow
(1) CBS
(3) ACM

The ALMA is implemented as an I/O event-driven process. The messages are handled in the order they come in. Communication with the CBS system is handled by the MI utilities. The MI utilities handle all input to the application and invoke the function associated with the message type. Output to the ACM (ALSP Common Module) is handled by the application code. The application code can be broken up into three functionally distinct areas: CBS message processing; ALSP and CBS object mapping; and ALSP message processing.

5 ALSP PARSER

The ALSP message-parsing portion of the ALSP application was implemented independently. The challenge was to quickly and effectively develop a parser that could handle the protocol for the messages described in the *ALSP Operational Specification*. The approach was to develop a top-level description of the ALSP protocol grammar. Figure 2 presents a notation of

extended BNF (Backus-Naur Form) used to describe the ALSP grammar, excluding the non-terminals STRING, INTEGER, FLOAT, and ENUMERATED. With these grammar rules, a portable (VAX/VMS and UNIX) ALSP parser was built using YACC and LEX. The parser creates a dynamic data structure in the format of a binary parse tree. The parse module is given an input string, and it produces either a pointer to a tree or null to indicate a syntax error. This tree represents a complete ALSP message; thus, given a pointer to the tree, other modules have easy access to all of the data associated with an ALSP message.

```
command   := message   NEWLINE

message   :=   Funct {Funct}

Funct     := KEYWORD LPAREN [Arglist] RPAREN

Arglist   := Arg { [ COMMA ] Arg }

Arg       := Funct  | Value  | Attribute

Attribute := Value   ATSIGN ENUMERATED

Value     := STRING INTEGER FLOAT ENUMERATED

ATSIGN    := '@'

LPAREN    := '('

RPAREN    := ')'

COMMA     := ','
```

Figure 2: Top-Level Description of ALSP Grammar

The parser was built independent from any semantic actions so that future changes and additions to ALSP keywords and enumerations would not require modifications. For example, with the ongoing implementation of the CBS-CSSTSS ALSP interface, new ALSP keywords with the associated argument type and list are included in a data file used by the ALSP verifier portion of the application. The parser required no changes. The verifier code provides an optional pass used to enforce the ALSP syntax at the level described in the Interface Control Documents. Following the creation of the parse tree, the verifier is handed a pointer to it. Similarly, the order generator is given a pointer to the tree and it outputs the CBS order. After the ALSP parser was defined, the next item on the agenda was to establish time synchronization between CBS and the confederation.

6 TIME SYNCHRONIZATION

As a member of the ALSP confederation, CBS had to give up absolute control over timing. Most of the difficulty was encountered in getting the CBS controller to think "confederation." In other words, the controller could no longer regulate the pace of the simulation without affecting the confederation. The ability to obey an external time monitor was facilitated with the new timing routine introduced in CBS 1.4. The CBS Executive uses the SIMSCRIPT global variable TIMESYNC.V to specify a user-defined time synchronization routine that is invoked before time is advanced. The CBS version of this routine yields the time to which the simulation will next advance. When part of a confederation, this routine yields the most recent time granted by the confederation. In order to coordinate timing with the confederation, a new message type was developed: CBS Control Message.

The ALMA conveys grant advances to CBS and receives advance requests from CBS via control messages. When the Executive receives a grant advance control message, it invokes a FORTH routine that saves the time to which the simulation will advance, and schedules the next advance request control message to be sent to the ALMA. This message triggers the application to send to the ACM an *advance_request* message. Two new FORTH commands were added to the CBS Executive to allow it to pause and resume the simulation when joined to the confederation.

7 DATA SYNCHRONIZATION

Saving and recovering simulation data are also coordinated between CBS and the ALMA via control messages. CBS did not have to change the way it saves data. A *start_save* initiated from the ACM will cause the ALMA to send a control message to the Executive. A save request control message will prompt the CBS controller to enter the preexisting CBS SAVE command, which performs a standard checkpoint. Synchronizing data during recovery, however, is more complicated. The burden of synchronizing ALSP objects after a rollback in time or a crash recovery, where history might change, belongs to the ALMA.

The ALMA is responsible for keeping CBS informed of the state of ALSP objects it is interested in "ghosting." Similarly, the ALMA keeps the confederation informed of the CBS simulation objects and events in which it is interested. To accomplish this, the ALMA keeps internal hash tables that are used to store the mapping between CBS objects and ALSP objects. Data may be lost when CBS or the ALMA crash, or when communication is interrupted between

CBS and the ALMA or between the ALMA and the ACM. The ALMA depends on the MI database for all object state data. After a CBS recovery, the MI database is refreshed, thus reflecting the latest state of simulation objects. The ALMA then, depending on the type of recovery, may rebuild its internal tables from scratch, or it may reconcile the differences between the objects in its tables and the objects in the MI database, simultaneously sending out *registers*, *updates*, and *deletes* to the confederation as needed.

8 MAPPING OF SIMULATION DATA

The ALMA keeps three different mappings. For ALSP objects, it establishes a one-to-one correspondence between their attributes in CBS and their ALSP attributes. When the ALMA receives change game truth for an object in which the confederation is interested, it checks to see if any of the changed attributes have a counterpart in the confederation. If so, an update message is generated. The second type of mapping involves mapping CBS objects to ALSP objects. Given the ALSP id, the CBS id, or the name of an object, the mapping routines will return a pointer to that object. The third mapping involves mapping ALSP terminology (enumerations) to CBS terminology. A scaling factor may also accompany a mapping. For example, many air-to-ground munitions in AWSIM map to a particular munitions type in CBS with a multiplying factor applied to the amount. All these mappings are needed to support CBS and ALSP message processing. ALSP *actor_to_acm_messages* are triggered from the game truth and event messages received from the CBS model.

9 THE FUTURE OF CBS AS AN ACTOR

As a result of the CBS-AWSIM and CBS-CSSTSS interfaces, CBS is emerging as the actor who will furnish the confederation with all unit *updates*. Since AWSIM and CSSTSS involve different functional areas, CBS can easily interface with both simultaneously. With the advent of more actors, it is clear that for each functional area of war simulation, a dominant actor must take control. It is easier to let an actor play its role in the confederation independent of other actors. Otherwise, every ALSP actor needs to be functionally aware of all other actors from a modeling standpoint. AWSIM should be able to interface with CBS independent of CSSTSS in supplying the missiles and launchers, for example, to CBS.

The initial definition of the functional interface between actors is very important. The interface control documents provide the only common development tool for the actors. As for any future work in the CBS translator, minor changes are foreseen compared to any functional changes that the model might need to accommodate. In conclusion, the ALSP protocol facilitates distributed war modeling, and integrates the functionality at the confederation level. CBS has successfully joined the ALSP confederation by defining its modeling role in the confederation and by accurately relaying its simulation data using the ALSP protocol.

ACKNOWLEDGMENTS

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the U.S. Army Simulation, Training, and Instrumentation Command, through an agreement with the National Aeronautics and Space Administration. Technical guidance was provided by the U.S. Army National Simulation Center. The author thanks Eric A. Tailor of the Jet Propulsion Laboratory for his contributions on the ALSP parser, and Martin Orton, a simulation expert contracted by the Jet Propulsion Laboratory, for his contributions on the CBS Executive.

REFERENCES

The MITRE Corporation. 1993. *Aggregate Level Simulation Protocol Technical Specification.* McLean, Virginia: The MITRE Corporation.

CACI Products Company. 1989. *VAX/VMS SIMSCRIPT II.5 User's Manual.* Release 5.1. La Jolla, California: CACI Products Company.

Jet Propulsion Laboratory. 1993. *CBS-AWSIM Interface Control Document.* JPL Internal Document D-9828, Rev. A. Pasadena, California: Jet Propulsion Laboratory.

Jet Propulsion Laboratory. 1993. *Corps Battle Simulation Version 1.4 Executive Overview.* JPL Internal Document D-7848, Rev. A. Pasadena, California: Jet Propulsion Laboratory.

AUTHOR BIOGRAPHY

LYDIA P. DUBON is a member of the technical staff for the Corps Battle Simulation project at the Jet Propulsion Laboratory, Pasadena, California. She received a B.S. degree in Computer Science and Engineering from the University of California at Los Angeles in 1987.