Management of the Joint Training Confederation Family of Specifications

Dave Prochnow
The MITRE Corporation
Mail Stop W647
11493 Sunset Hills Road
Reston, VA 22090
prochnow@mitre.org

Ernest Page
The MITRE Corporation
Mail Stop W647
11493 Sunset Hills Road
Reston, VA 22090
epage@mitre.org

Dr. Mary C. Fischer
US Army Simulation, Training, and Instrumentation Command
AMCPM-FAMS (Dr. Fischer)
12350 Research Parkway
Orlando FL 32826-3276
fischerm@stricom.army.mil

Keywords

High Level Architecture, HLA, Joint Training Confederation, JTC, Federation Object Model, FOM, Simulation Object Model, SOM, Object Model Template, OMT

Abstract

The High Level Architecture (HLA) stipulates model specifications in the form of Federation Object Models (FOMs) and Simulation Object Models (SOMs). These specifications are constructed in accordance with the Object Model Template (OMT) and establish the "information model contract that is necessary (but not sufficient) to ensure interoperability among federates" [4]. Predating the definition of the HLA, the Aggregate Level Simulation Protocol (ALSP) Joint Training Confederation (JTC) utilizes a family of specifications that in many ways is similar to those defined for the HLA. The JTC is an evolving confederation of models, and the evolution and maintenance of the JTC specifications demands equal consideration to that of the underlying code.

The ALSP JTC family of specifications is described and compared with respect to both content and format to the requirements stated in the HLA. The processes supporting the evolution and maintenance of the JTC specification are presented, and future initiatives briefly described.

1.0 Introduction

The Aggregate Level Simulation Protocol (ALSP) is a multi-Service program with the US Army Simulation, Training, and Instrumentation Command (STRICOM) as the Department of Defense (DoD) Executive Agent. ALSP consists of the software and protocols that permits disparate simulations to work together as an integrated whole. An ALSP *confederation* consists of multiple wargames, each sharing information with the others. They communicate through an established protocol that describes the objects to be shared and the interactions among these shared objects. A collection of system-level software has been developed that facilitates intra-confederation communication, and coordinates confederation issues such as time advance and object ownership. For details of the ALSP Infrastructure Software (AIS) refer to [11].

The Joint Training Confederation (JTC) is the largest application of ALSP. The JTC has been used since 1992 to train military officers all over the world, including the United States, Germany, Korea, and Japan. The 1997 JTC includes twelve *actors* (i.e., simulations) which provide specific functionality to support the intended training audience. These simulations and their respective roles within the JTC are illustrated in Figure 1.

The JTC is an evolving system. This paper addresses the complexity of developing and managing the evolution of the JTC specification, focusing primarily on the information management software and underlying methodology. We review the JTC specifications in Section 2 and compare these to the specifications defined for the High Level Architecture (HLA) in Section 3. In Section 4, we describe the evolution and current management strategy for the JTC specifications. Section 5 provides conclusions and recommendations.

2.0 The JTC Specifications

Similar to the HLA, the fundamental components of the JTC specifications are *objects* and *interactions*. Currently, the JTC defines 22 object classes representing military entities. Each object class is described by a set of *attributes*, and each simulation in the JTC broadcasts and receives *updates* on attributes that are relevant for its model. Additionally, when an object in one simulation directly affects an object in another simulation, an *interaction* message is passed between them. The JTC identifies 66 different types of interactions. Each interaction is described by a set of *parameters*. The implementation of objects and interactions differs among JTC actors. Thus, the JTC specifications consist of information regarding actors, objects, attributes, parameters, and interactions, and the relationships among them.

2.1 The JTC Objects

A JTC public object is an entity about which information is exchanged between the simulations in the JTC through the AIS. Each simulation may have additional objects which are private to that particular simulation and about which information is not exchanged with the other simulations in the JTC.

The class structure for the ALSP JTC is hierarchically organized with four primary domains: air, ground, sea, and space. Lower levels reflect the implementation of specific functionality in the JTC. For example: AIR.FIXEDWING, GROUND.MANEUVER.COMBAT, SEA.SURFACE.SHIP, and SPACE.SATELLITE. A full listing of JTC object classes can be found in the <u>Aggregate Level Simulation Protocol (ALSP) 1997 Joint Training Confederation Operational Specification</u> [8].

2.2 JTC Object Attributes

A total of 75 attributes currently describe the JTC state space. These attributes are of varying data types, including string, numeric, composite[1], and enumerated. The values for enumerated data types are determined by agreement among JTC constituents. Each JTC object is defined by a subset of the 75 attributes. Several attributes are common across all JTC objects including, for example, attributes that provide object identification, affiliation and location. Most attributes are specific to a particular class of object. For example, only air objects have an altitude attribute. For brevity,

[1] A composite attribute consists of two or more attributes, which can each be of string, numeric, composite or enumerated data types.

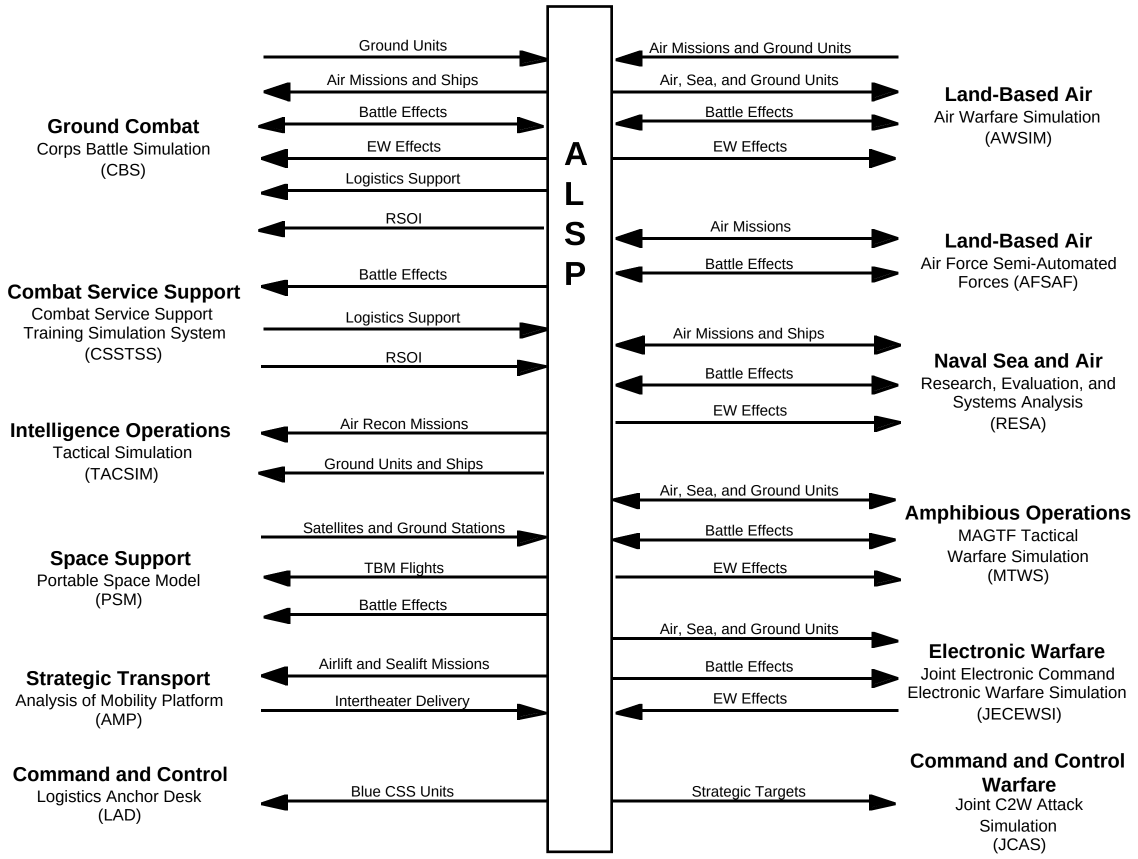

Figure 1. Operational View of the 1997 JTC

neither the JTC attributes nor their definitions are listed here. They appear in [8].

An actor that represents instances of a public object is *not* required to represent all of the attributes defined for that object class. An actor may either update or reflect (using HLA terminology) subsets of object attributes. An indication occurs at runtime through registration information sent to the AIS at the beginning of confederation execution. Although this data is provided to the software, the identification and specification of these subsets must occur well before JTC use; updating (and broadcasting) attribute values that are not required by other actors wastes resources. Potentially more damaging is the case where an actor requires an attribute in order to ghost[2] a particular object, but no actor is providing updates for the attribute.

2.3 JTC Interactions

When actions in one actor directly affect one or more other actors, an interaction message is passed across ALSP. Currently, the JTC employs 66 different interactions. Examples are combat engagements, transfers of assets, and orders.

[2] *Ghosting* refers to the ability to internally maintain information on an object owned by another simulation. An actor receives updates on ghosted objects and might interact with them. Ghosting in ALSP is analogous to reflecting in HLA.

The JTC paradigm for combat between entities in different actors is that the actor with the attacking object sends an engagement interaction which is received by the actor with the target object; in turn, the target actor responds with a report interaction to indicate how many of its assets were killed and damaged. Similarly, other functionality across simulations is achieved through a combination of interaction messages (a dialogue). Again, for brevity, the different kinds of interaction messages in the JTC will not be listed here.

2.4 JTC Parameters

In ALSP, parameters are to interactions as attributes are to objects. That is, each interaction is composed of parameters. Currently, there are about 50 parameters used in the JTC. Similar to attributes, each parameter has a name, definition, data type, and, if applicable, a list of enumerated values. If a parameter uses enumerated values, then there must be agreement among JTC constituents on the enumerated values.

The basic parameters of an interaction are "to", "from", and "parameters". The to and from parameters indicate the initiating object and the affected object, respectively. It is not required that each interaction type have a to and from parameter, but they must have at least one of these. The "parameters" parameter is actually a composite parameter, containing the other parameters associated with the interaction. For instance, an air-to-ship engagement interaction contains parameters for the latitude, longitude, and salvo. Salvo is a composite attribute, and it contains combinations of weapon types and quantities involved in the engagement.

2.5 Summary of JTC Specifications

For all objects, attributes, interactions, parameters, and actors in the JTC, there must be significant coordination among simulation developers on their specifications. Additionally, the specifications must detail the relationships between:

- Actors and Objects
- Actors and Attributes
- Actors and Interactions
- Objects and Attributes
- Interactions and Parameters
- Attributes and Enumerated Values
- Parameters and Enumerated Values

As explained earlier, this coordination must occur well before any confederation execution. Otherwise, actors will send irrelevant data and receive undesired data.

2.6 The JTC Specifications Document Set

The ALSP community maintains several documents for JTC specifications information. These documents, which are updated annually, are listed and briefly described below:

- **Joint Training Confederation Operational Specification.** The JTC operational specification document contains the description of the JTC object model. This document contains tables for objects owned or ghosted by each actor, attributes updated or ghosted for each object class by each actor, interactions sent and received by each actor, and parameters contained within each interaction. The JTC operational specification details the static aspects of the JTC specification and is equivalent to the HLA Federation Object Model (FOM).

- **Actor Implementation Documents (AIDs).** Each simulation developer, prior to the testing phase of a confederation year, submits an AID which describes how its actor will be used in the JTC. In addition to textual information, each AID has tables indicating which objects are owned or ghosted by the actor, which attributes of specified objects are updated or received, and which interactions are sent or received.

- **Interface Control Documents (ICDs).** The ICDs describe the exchange of messages between actors in order to simulate various activities where the modeling is achieved with more than one simulation. An ICD describes the dynamic (behavioral) aspects of the JTC specification. The JTC contains an ICD for each of four functional areas: combat engagements, electronic warfare, logistics, and entity-level air modeling.

- **Enumerations List.** The enumerations list contains the allowable values for all JTC enumerated attributes. This permits all JTC actors to have a common understanding of allowable names.

All of these JTC documents are maintained on a common server and, as a result, are accessible by all members of the JTC community.

3.0 HLA Specifications and How They Compare with the JTC Specifications

The DoD, in its Modeling and Simulation Master Plan [3] mandates that future simulation systems use the High Level Architecture (HLA). The HLA, currently under development, will support the interoperability of simulations, or *federates*. The backbone of HLA is the Runtime Infrastructure (RTI), the software which allows the data exchange between simulations in an HLA federation. The HLA consists of three components:

- HLA Rules
- HLA Interface Specification
- HLA Object Model Template (OMT)

The HLA OMT v. 1.0 document [4] describes a Federation Object Model (FOM). The goals and content of the FOM are almost identical to those for the JTC operational specification. The OMT also describes Simulation Object Models (SOMs). SOMs share the same structure as FOMs but are designed to document the intrinsic (federation independent) capabilities of a federate. There is no analogous documentation within the JTC; all of the JTC documentation is federation oriented.

Table 1 below compares the OMT information and structure with that of ALSP. As the table indicates, both the HLA and ALSP have much in common. One difference, however, is that in ALSP, actors can define a minimum set of attributes (called a *create* set) needed to create a ghost (reflection) of an object. A similar concept may be useful in the HLA. Otherwise, a federate is forced to actively request attribute values needed to construct a reflection rather than allowing the infrastructure to accomplish the task.

A few differences in the handling of interactions in HLA and ALSP are evident. In HLA, a federate must explicitly register an interaction for publication before it can broadcast that interaction. The OMT defines the initiating and affected object classes associated with each type of interaction. These features are not present in ALSP. On the other hand, the HLA currently includes no mechanism to describe the exchange of interactions that are frequently necessary to achieve certain functionality. For example, if an engagement interaction exists to fire artillery into a specified area (with some target objects that the initiating federate may not even know about), the engagement is not complete without knowing what damage was caused by the munition. This might require an interaction (or update) to be returned by the affected federates. Another example would be a logistics simulation which sends an interaction to order a convoy to be moved in another simulation. If the other simulation cannot move the convoy (e.g., a road or bridge is destroyed), the logistics simulation needs to receive notification to enact contingency plans for the delivery of supplies. Again, this would most likely be accomplished with another interaction message. In ALSP, these types of exchanges are defined in Interface Control Documents (ICDs). Similar provisions for the expression of model dynamics should prove useful for HLA.

OMT Structure Element	HLA	ALSP
Object class hierarchy	4	4
Object class names in ASCII	4	4
Object class definitions	4	4
Inheritance of subclasses from superclasses	4	4
Object class registration	4	4
Object attribute registration	4	4
Object class subscription	4	4
Object attribute subscription	4	4
Definition of minimum attribute set for receiving object data	6	4 [3]
Object attribute definitions	4	4
Attribute enumerations	4	4
Attribute complex data types	4	4
Interaction hierarchy	4	4
Interaction names in ASCII	4	4
Interaction Definitions	4	4
Inheritance of sub-interactions from super-interactions	4	4
Interaction registration	4	6 [4]
Interaction subscription	4	4
Initiating object class for interactions	4	6 [5]
Affected object class for interactions	4 [6]	6
Interaction dialogue	6	4 [7]
Interaction parameter definitions	4	4
Parameter enumerations	4	4
Parameter complex data types	4	4

Table 1. Comparison of OMT Structure in HLA and ALSP.

4.0 Management of the JTC Specifications

In traditional settings, software specifications are used to: (1) establish expectations and requirements, (2) guide development, and (3) evaluate the end product. Once the software is released the specification is rarely consulted, except to the extent that aspects of the specification are incorporated into the software user's documentation. Within the JTC, the specifications are often vital resources for exercise support. Configuration management within the JTC is a difficult proposition – JTC members are typically Service simulations with *primary* roles outside of the JTC. Coordination of twelve, independent, primary development schedules with the JTC development schedule is extremely difficult. The voluntary nature of JTC participation also complicates the generation of fixed, useful JTC software baselines. Although strong efforts are made to limit the scope and nature of "post-delivery" software modifications, the reality of the situation is that software changes occur year-round – and quite often political considerations are paramount to the technical realities. Therefore, in time-critical situations like actual JTC exercises, rapid access

[3] In ALSP, this minimum set of attributes required for ghosting objects is referred to as the create set.
[4] In ALSP, actors can send interactions, but there is no prior registration required.
[5] There are initiating and affected object classes in ALSP, but they are not formally defined in the specifications.
[6] The HLA OMT further subdivides the affected object classes into those that sense the interaction and those that react to the interaction.
[7] The exchange of interaction messages are defined in the Interface Control Documents (ICDs).

to the specification is valuable in order to quickly resolve a wide array of problems that can (and do) arise.

As indicated in Section 2.0, the JTC object model consists of actors, attributes, objects, parameters, and interactions, as well as the intersection among these. Each actor determines which objects it will own or ghost, and the attributes for which it will send or receive updates, and the interactions it will send or receive.

Using a bottom-up approach, the smallest items to be represented in the JTC object model are attributes and parameters. Each object and parameter is defined with four basic elements: a name, a definition, a data type, and if applicable, a list of enumerated values. The enumeration list is only applicable for enumerated data types.

The next higher-level items for the JTC object model are the simulated object classes and the interactions. Each object class is composed of attributes, and each interaction type is composed of parameters.

The highest-level items in the JTC object model are the simulations, or actors. Each actor defines which objects and which interactions to employ in its implementation. Actors determine which objects they will own and/or ghost and which interaction types that they will send and/or receive. For the simulated object classes, an actor also specifies the individual attributes that it will update or receive. For each object class that an actor owns, it defines an "update set", which is the set of all attributes for which it will provide updates to the rest of the confederation. If an object is ghosted by an actor, then both a "create set" and an "interest set" of attributes are defined. The create set defines the minimal set of attributes required for an object to be ghosted. Thus, if any of the attributes in the create set are not provided by other actors, then a ghost object is never created for the actor. The interest set defines optional attributes for a ghosted object. That is, an actor will use values for interest attributes if they are provided, but if not provided, then the actor uses default values.

4.1 Previous JTC Object Model Management

Before discussing the previous process for object model management in the JTC, some background is necessary. In 1990, the Defense Advanced Research Projects Agency (DARPA) engaged The MITRE Corporation to study the application of distributed interactive simulation (DIS), then simulation network (SIMNET), principles to aggregate-level, constructive training simulations. Based on prototype efforts, a community-based experiment began in 1991 to develop protocols and software to link the U.S. Army's Corps Battle Simulation (CBS) and the U.S. Air Force's Air Warfare Simulation (AWSIM). The success of the prototype and the users' recognition that this technology met their exercise requirements rapidly led to production-quality software. The first ALSP confederation, supporting air-ground interactions between CBS and AWSIM, supported three major exercises in 1992.

The first confederation had only two actors. The object model was fairly simple. Updates or interactions broadcast by one actor were received by the other actor. Coordination is still required between the two simulations, but this activity is straightforward. As the Joint Training Confederation has evolved, new actors have entered the confederation each year. In 1997, 12 actors are projected for JTC membership. MITRE, the ALSP systems engineer, has maintained the object model in an annual document called the Aggregate Level Simulation Protocol (ALSP) 19xx Joint Training Confederation Operational Specification, where "xx" represents the year of the document. In order to produce this document, MITRE oversees the collection of data from each actor regarding their create, update, and interest sets for each object class, as well as on the interactions that they will send or receive. This is typically a labor-intensive process, requiring numerous interactions with a variety of actor representatives[8].

A few years ago, a new requirement was levied on each actor to produce an Actor Implementation Document (AID). The data in

[8] An actor usually has both a developer and a proponent agency, and many times both of these must be contacted to elicit operational specification information.

the AIDs could then be used to produce the Operational Specification document. Once the object model is fully documented, MITRE places the operational specifications on a server, accessible by all actors and simulation sites, for their review and comment.

As indicated in Section 2.2, the creation of an object model occurs prior to testing. In many cases, changes in the implementation of an ALSP interface by one actor directly affects the implementation requirements of one or more other actors in the confederation. In the JTC, there have been several cases in which an actor cannot fully implement a planned interface because of late-occurring technical or funding circumstances. Such changes require other actors to modify their software for a partial implementation of a concept, or to "undo" software modifications when the other actor cannot implement the concept at all.

The process described in the preceding paragraphs worked well when not much more than a handful of actors were in a confederation. However, with the increasing size of the JTC, this process has proven difficult for successful coordination among all actors. The next section describes the new JTC process for managing its object model.

4.2 New JTC Object Model Management

With increase in scope of the JTC, management of the JTC object model is becoming increasingly difficult. For the proposed 1997 JTC, data had to be collected from 12 different actors to create the operational specifications document. This process is time-consuming and error-prone and requires persistent (if not always polite) queries to the myriad sources of information. And since each information source is primarily familiar with the modifications being made to its actor, the affect on other actors is not always realized or communicated. Thus, anomalies arise that must be detected and properly reconciled.

To provide automated object model analysis, MITRE developed the JTC Operational Specifications Software (JOSS). JOSS is a Java program that reads the object model data, stores it into Java classes (for actors, simulated objects, attributes, interactions, and parameters), analyzes the data, and produces a variety of HTML pages that describe the object mode and the analysis results. Currently, JOSS reads data from a pre-formatted comma-separated value (CSV) file. Using a CSV file allows the data to be easily loaded into a spreadsheet for analysis or modification. Furthermore, since most spreadsheet programs can read and write CSV files, the JOSS is relatively spreadsheet-independent.

JOSS produces HTML cross-references for the actors, objects, interactions, attributes, and parameters in the JTC object model. An HTML page exists for each actor, each object, and each interaction. The attributes and parameters are defined on single pages. The actors' HTML pages summarize objects owned and ghosted, and interactions sent and received, with links to the associated objects, attributes, and interactions. The objects' HTML pages summarize the attributes and how they are used by each actor. Thus, they have links to the associated actors and attributes. Furthermore, the interactions' HTML pages summarize the parameters and the actors which send or receive the interaction. Finally, each actor, object, and interaction page has links to the other actors, objects, and interaction pages, respectively. Figure 2 shows all the links between the actor, object, interaction, attribute, and parameter HTML pages.

In the example shown in Figure 2, Actor A owns or ghosts Object X, so it has a link to that object's page. Actor A does not send or receive neither Interaction I nor Interaction J, so it has no links to these interactions. Actor B owns or ghosts both Objects X and Y, in addition to sending or receiving Interaction I. Actor C owns or ghosts Object Y and sends or receives both Interactions I and J. Note that the object and interaction pages refer back to their associated actors. Also note that all object pages and all actor pages link to the attribute definition page, and that all interaction pages link to the parameter definition page. Although not shown in the picture, each page also has a link back to the main JTC operational specifications page, which is described below.

In addition to the pages on individual actors, objects, interactions, attributes, and parameters, JOSS produces pages with summary information. It generates one page with a summary of actors and objects, showing a large table of objects owned and ghosted by each actor. JOSS also outputs an interaction summary page, with a table of interactions sent and received by each actor.

As indicated above, JOSS has an analytical capability. JOSS determines which attributes are being updated by an actor but then never ghosted by another actor, and more importantly, which attributes that an actor deems essential for a ghost object but which are never updated by an actor.[9] If any anomalies are identified for an object class, then they are added to the object's HTML pages. There also exists a summary of all JTC anomalies on a separate, special HTML page.

Finally, JOSS produces a main page for the JTC operational specifications. At the top of the page, it contains links to the object summary page, the interaction page, and the anomaly page. It also contains menus of links to each actor, object, and interaction. All JOSS output resides on a web server accessible by all JTC developers. The systems engineer alerts the community, via an automated list server, that this data is available for review and comment. Developers then analyze the data themselves and determine what changes must be made in their simulations. Any changes are directed back to the systems engineer, who then modifies the CSV file, re-executes the JOSS, and then replaces the HTML pages on the server. The turnaround time is less than 30 minutes, so the users have access to a dynamic object model.

JOSS also has an additional capability of producing data that can be used for testing. JOSS produces files on object class data and interaction data that can then be fed into a "test harness" tool. This utility is used to test ALSP translators by generating creations and updates of objects based on the associated attributes, their data types, and, if applicable, their list of enumerated values. Similarly, the test tool will output interaction messages with the relevant parameters and data types.

[9] The Confederation Reconciliation Tool (CRT) produces similar cross-references based on runtime actor registration data.

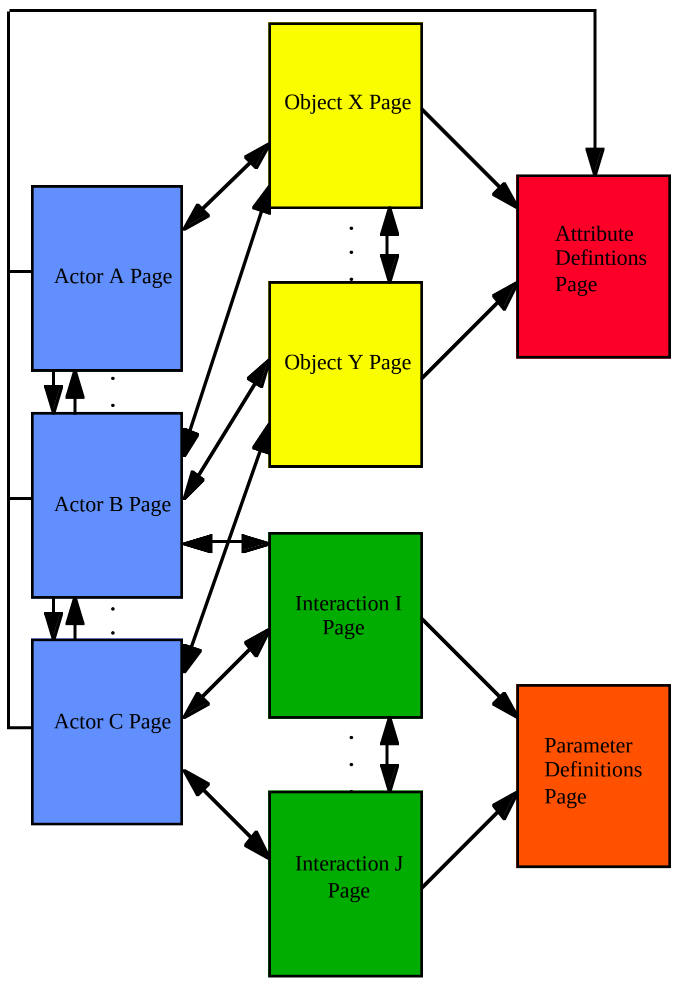

Figure 2. Example of Cross-Linking in JOSS HTML Pages

4.3 Other Improvements Envisioned for the JTC Object Model

While the current JTC object model is now maintained in spreadsheets, a transition to commercial databases is envisioned. Furthermore, we would like to make this data accessible to all users in the ALSP community. One possibility is to use new commercially developed web-based tools which allow users from remote sites to access database information stored at a particular site. Taking these steps would simplify the maintenance of the JTC object model for the systems engineer and provide better service to the JTC developers. The JOSS program would still be used to perform analysis of the operational specifications data and to produce relevant HTML pages. The only difference is that the JOSS would extract data from the commercial database, rather than from the CSV files.

5.0 Conclusions

Predating the definition of the HLA, ALSP has evolved a set of specifications very similar in nature to those stipulated in the HLA Object Model Template. Experience with the JTC has shown that the model specifications can play a valuable role in exercise support, and that automated mechanisms for specification development, analysis, storage and retrieval are critical. The JOSS is an example of such a mechanism.

Experience with ALSP has also shown that representing the dynamic, *behavioral* aspects of a model is equally critical to the specification process as capturing static model characteristics. Included in this latter category are class and interaction hierarchies, object-attribute and interaction-parameter associations, and publication and subscription information. The OMT provides ample mechanisms for capturing static model information, but lacks clear guidance or language for the description of model dynamics. Within the JTC, Interface Control Documents (ICDs) serve in this capacity. Currently oriented around functional interfaces, e.g. air-to-ground combat, alternative orientations are being investigated (e.g. process orientation). The discrete event simulation community has been actively investigating simulation model specification for over 30 years. Simulation model specification languages (e.g. DELTA [7], SMSDL [6]) or other formalisms for model description (e.g. control flow graphs [2], DEVS [12], UNITY [1]) that *explicitly* support the expression of model dynamics should be considered as candidates for the HLA.

Acronyms

AFSAF	Air Force Semi-Automated Forces
AID	Actor Implementation Document
AIS	ALSP Infrastructure Software
ALSP	Aggregate Level Simulation Protocol
AMP	Analysis of Mobility Platform
AWSIM	Air Warfare Simulation
CBS	Corps Battle Simulation
CSSTSS	Combat Service Support Training Simulation System
CSV	Comma-Separated Values
FOM	Federation Object Model
HLA	High Level Architecture
HTML	Hypertext Management Language
ICD	Interface Control Document
JCCWSS	Joint Command and Control Warfare Simulation System
JTC	Joint Training Confederation
JOSS	JTC Operational Specification Software
JOVE	Joint Operational Visualization Environment
LAD	Logistics Anchor Desk
MTWS	MAGTF Tactical Warfare Simulation
OMT	Object Model Template
PSM	Portable Space Model
RDBMS	Relational Database Management System
RESA	Research, Evaluation, and System Analysis model
RTI	Runtime Infrastructure (for HLA)
SOM	Simulation Object Model
STRICOM	US Army's Simulation, Training, and Instrumentation Command
TACSIM	Tactical Simulation

Bibliography

[1] Chandy, K.M. and Misra, J. (1988). Parallel Program Design: A Foundation, Addison-Wesley, Reading, MA.

[2] Cota, B.A. and Sargent, R.G. (1990). “Control Flow Graphs: A Method of Model Representation for Discrete Event Simulation,” CASE Center Technical Report 9026, Syracuse University, Syracuse, NY, Nov.

[3] Department of Defense, Under Secretary for Defense (Acquisition and Technology), DoD Modeling and Simulation (M&S) Master Plan, Washington, DC, Department of Defense, October 1995.

[4] DMSO, Department of Defense High Level Architecture OMT Version 1.0, August 15, 1996.

[5] DMSO, Department of Defense High Level Architecture Rules Version 1.0, August 15, 1996.

[6] Frankowski, E.N. and Franta, W.R. (1980). “A Process Oriented Simulation Model Specification and Documentation Language,” Software – Practice and Experiences, 10(9), pp. 721-742, September.

[7] Handlykken, P. and Nygaard, K. (1980). “The DELTA System Description Language Motivation, Main Concepts, and Experience from Use,” In: Software Engineering Environments, H. HUNKE (Ed.), pp. 173-190, North-Holland, Amsterdam.

[8] MITRE Corporation, Aggregate Level Simulation Protocol (ALSP) 1997 Joint Training Confederation Operational Specification, September 1996.

[9] MITRE Corporation, Aggregate Level Simulation Protocol (ALSP) Executive Overview, February 1996.

[10] MITRE Corporation, Aggregate Level Simulation Protocol (ALSP) Joint Training Confederation Master Plan, October 1996.

[11]Weatherly, R., Wilson, A., Canova, B., Page, E., Zabek, A., Fischer, M., January 1996, Advanced Distributed Simulation Through the Aggregate Level Simulation Protocol, published in Proceedings of the 29th Hawaii International Conference on System Sciences, Volume 1, pp. 407-415, Wailea, Hawaii, 3-6 January, 1996.

[12] Zeigler, B.P. (1976). Theory of Modeling and Simulation, John Wiley and Sons, New York, NY.