Providing Conceptual Framework Support for Distributed Web-Based Simulation within the High Level Architecture

Ernest H. Page, Sean P. Griffin and S. Lynn Rother

The MITRE Corporation, 1820 Dolley Madison Blvd., McLean, VA 22102

ABSTRACT

Web-based simulation, a subject of increasing interest to both simulation researchers and practitioners, has the potential to significantly influence the application and availability of simulation as a problem-solving technique. Web technologies also portend cost-effective distributed modeling and simulation. These applications will require solutions to the systems interoperability problem similar to the DoD High Level Architecture (HLA). The suitability of the HLA to serve "mainstream" simulation is examined. Approaches for incorporating discrete event simulation conceptual frameworks within the HLA are described and ongoing research in this area noted. Issues raised include a discussion of the appropriate roles for a simulation-support language and a simulation-support architecture.

Keywords: Conceptual frameworks, distributed simulation, High Level Architecture, web-based simulation

1. INTRODUCTION

Web-based simulation is an emerging area of investigation within the simulation community. While the term "web-based simulation" can be used to describe many things,[1] it is typically used to denote the invocation of simulation programs over the Internet, specifically through a web browser.

The U.S. Department of Defense (DoD) High Level Architecture (HLA) has been mandated as a standard for all DoD modeling and simulation (M&S) efforts.[2] The HLA provides a common framework within which the runtime interoperability of disparate simulations is fostered. Through the HLA, the DoD is seeking to maximize the benefits of its M&S expenditures by encouraging the broadest application of these models and simulations.

On the surface, web-based simulation and the HLA may seem to have little in common. But their commonality becomes evident when the notion of *distribution* is considered. In one sense, all of web-based simulation is distributed simulation. Through a browser interface, a user invokes a simulation residing on a remote computer – a basic application of the client-server model of computing. With the code mobility feature of Java, the simulation can be made to execute on either the client or the server, but it is still basically client-server computing. However, with the proliferation of web-based technologies and an increasing availability of simulations[*] on the web, a *cultural demand* for techniques that permit the integration of tools and models has been predicted.[3] That is, the culture of the web will drive the academic and commercial sectors to seek a solution to the interoperability problem that the DoD has confronted with the HLA. Web-based simulation may become synonymous with distributed modeling and simulation.

So the question naturally arises, is the HLA a suitable technology for distributed (web-based) simulation in mainstream, i.e. non-DoD, computing? Acceptance of the HLA in this role may depend, at least in part, in the ability of the HLA to support the methods for model development, analysis and execution that are commonplace within the mainstream. In this paper we briefly examine one aspect of such support: the utilization of discrete event simulation conceptual frameworks within the context of the HLA. We describe the means by which this support may be effected, and note areas of current research and development. We consider the question of the proper separation of roles between a simulation-support language and a simulation-support architecture. Our presentation of background material is limited. The reader is assumed to possess a working familiarity with the basic concepts of discrete event simulation and DoD M&S. Interested readers are encouraged to consult the references for detailed treatments of supporting concepts.

This article is to appear in: *Proceedings of SPIE: Enabling Technologies for Simulation Science II,* Orlando, Fl, 13-17 April 1998.

[*]We adopt a broad definition of simulation here that includes gaming applications.

2. THE HIGH LEVEL ARCHITECTURE

The High Level Architecture (HLA) has been proposed and developed to support reuse and interoperation of simulations across the U.S. Department of Defense.[4] The HLA represents a generalization and extension of the Distributed Interactive Simulation (DIS) protocols[5] and the Aggregate Level Simulation Protocol.[6–8] The HLA is defined by three components:

- a common model definition and specification formalism;[9]
- a collection of services describing the HLA runtime environment;[10]
- a set of rules governing compliance with the architecture;[11]

and is intended to have wide applicability across the full range of defense simulation applications, including training, analysis and engineering, and at various levels of resolution.

At the center of the HLA is the notion of a *federation.* A federation is a collection of simulations and other systems – *federates* – that interoperate using the protocols described by the architecture. A Federation Object Model (FOM), constructed in accordance with the required formalism,[9] provides the model specification and establishes a contract between the federates regarding the nature of activity taking place during federation runtime. Model execution is accomplished through an HLA Runtime Infrastructure (RTI) which is an implementation of the infrastructure services defined in Ref. 10.[†]

In a typical federation execution, federates join the federation, indicate their operating parameters (e.g. information they will provide *to* the federation and information they will accept *from* the federation) and then participate in the evolution of federation state until the federate departs the federation, or the simulation terminates. FOM data is provided to the RTI at runtime, enabling the infrastructure to provide a level of enforcement with respect to the information contract.

A flexible time flow mechanism has been defined for the architecture, permitting a wide range of implementations from tightly synchronous, causality preserving, fully-reliable interprocess communication to completely asynchronous, non-timestamped, unreliable interprocess communication, and all points in between.[12]

The HLA was conceived and developed to be a framework for distributed simulation within the DoD, but with a high degree of generality in several key areas, the HLA is potentially a suitable framework for many types of distributed enterprises. For complete descriptions of the HLA, it's motivations and components, and for access to prototype software, consult the HLA web site, `http://hla.dmso.mil`.

3. INCORPORATING CONCEPTUAL FRAMEWORK SUPPORT WITHIN THE HLA

A *conceptual framework* (CF) is an underlying structure and organization of ideas that constitutes the outline and basic frame which guide a modeler in representing a system in the form of a model.[13–15] Also referred to as a *simulation strategy* or *world view,* the common discrete event simulation CFs are briefly reviewed below.

3.1. Conceptual Frameworks for Discrete Event Simulation

Lackner[16] gives the first characterization of conceptual frameworks. He makes the observation that the final expression of a digital simulation model is computer code – algorithms and data – but such a categorization of system elements does not produce a useful model. A more restrictive set of categories is necessary to the establishment of a general approach to modeling systems. Lackner asserts that such a restrictive set of categories is identified with a special view or apprehension of reality as a whole, a *Weltansicht,* which a modeler adopts when contemplating an object system, "A modeler looks at the system in a certain way; certain kinds of things are contemplated, and an orderly scheme of relationships among these kinds of things is assumed."

Kiviat[17,18] expands Lackner's development identifying three world views: event-oriented, activity-oriented and process-oriented. Fishman[19] further elaborates the set and suggests the labels that are most widely referenced today:

[†]In addition to defining services for the runtime infrastructure, the HLA Interface Specification defines services that must be implemented by federates.

- *Event Scheduling.* The model is described in terms of *events.* An event is a change in state that occurs at an *instant* in simulation time.[20] An event is said to be *determined* if the condition on its occurrence can be expressed strictly as a function of simulation time, otherwise the event is said to be *contingent.* In an event scheduling implementation, *event routines* encompass all the model attribute value changes that comprise a given event. Event routines associated with determined events can be scheduled on an *event list,* which is a timestamp-ordered collection of event notices. Contingent events are dependent on state variables whose values cannot be predicted in advance; checks for contingent events are contained within the event routines. Event scheduling languages include GASP, SIMSCRIPT, SIMAN and SLAM.

- *Activity Scanning.* The model is described in terms of *activities.* An activity is the state of the system (or any of its components) between two events.[20] A modeler identifies the objects in the system, the activities they undergo and the conditions under which these activities occur. Implementations of activity scanning models can adopt either a fixed-time increment (two-phase) or variable-time increment (three-phase) time flow mechanism.[13] Activity scanning languages, which have achieved greater popularity in the U.K. than in the U.S., include GSP, ECSL and OPS-4.

- *Process Interaction.* The model is described in terms of *processes.* A process is the contiguous succession of one or more activities.[20] Using a process view, a modeler identifies the objects in the system and describes their respective processes – the sequence of activities between object creation and object destruction. As an object moves through its process, it may experience delays. Delays can have a determined or variable duration. When an object experiences a delay, it becomes *passive,* and another object becomes *active,* initiating or resuming its process. Process interaction languages include Simula and MODSIM III.

- *Transaction flow.* Transaction flow is a variant of process interaction in which the logic of the model is described from the point of view of the *dynamic* objects in the model. Transaction flow models support the material-oriented view of systems described by Tocher.[21] Process interaction languages supporting the transaction flow view include GPSS/H and MODSIM III.

Derrick[22] identifies and characterizes a variety of alternative world views for discrete event simulation, but those identified above are predominant.

3.2. Models for Incorporation

The conceptual framework offered by the High Level Architecture is actually quite robust and expressive in its support for distributed enterprises, but at the same time the HLA does not offer the range of support for simulation model organization that is found within the traditional discrete event simulation CFs. In this area, the services defined in the HLA are fairly rudimentary. In some cases, the differences between the HLA CF and DES CFs are simply matters of vocabulary, and are thus easily reconciled. In other cases, however, DES CFs provide a level of abstraction for which there is no direct mapping in the HLA. For example, as noted in Section 2, the time flow mechanism within HLA is highly flexible, but its access methods – `timeAdvanceRequest, nextEventRequest, timeAdvanceRequestAvailable, nextEventRequestAvailable` – require direct invocation by a federate. These services are easily adaptable by simulations that directly manipulate their event list, like typical event scheduling and activity scanning models, but the type of indirect manipulation provided in process interaction models has no corollary within the HLA.

Can support for DES conceptual frameworks be incorporated within the context of the HLA? Absolutely. Avenues toward this incorporation include:

- *Language translation.* Currently, the HLA is not programmable using simulation programming languages (SPLs), that is, extant RTI prototypes require federates be constructed in general purpose languages such as, C, C++, Ada and Java.[‡] Adopting the language translation approach, a simulation programming language is used to develop the model; the SPL and supporting model specification formalism (if any) provide the CF support. The SPL-oriented representation is translated (automatically or semi-automatically) into an RTI-ready source language and embedded within the structure of a generic HLA federate.

[‡]It may be argued that this is a reasonable choice since relatively few DoD M&S applications are developed in SPLs. On the other hand, one could blame the lack of SPL use on failure of the DoD to support their use.

- *Language extension and integration.* This approach includes: (1) the extension of (RTI-ready) general-purpose languages to provide SPL-level functionality, and (2) modification of SPLs to provide direct support for HLA services within the language.
- *Architectural extension.* Using this approach, the HLA itself is extended through nomenclature evolution and the addition of services, as necessary, to support DES CFs.

The language translation approach has the advantage that it is potentially transparent to the simulation modeler. HLA integration is accomplished by embedding the original simulation within a generic federate shell. All of the domain knowledge required to use the HLA is encapsulated in this shell. The disadvantage of this approach is that the full range of HLA capabilities may not be exploited if the base SPL does not support them. For example, Klein, Straßburger and Beikrich[23] describe efforts to translate GPSS programs into a Java federate shell. The authors note that permitting modelers to effectively utilize the HLA facility for *interactions* in this approach remains an open question, since GPSS contains no equivalent notion or construct.

The primary advantage of the language extension and integration approach is that it *synthesizes* the CF of the language with the distributed operating system CF implied by HLA within the context of the (familiar) simulation language. The disadvantage of this approach is that not every general-purpose language is suited to accommodate an SPL role.[§] Without a suitable framework for the integration of simulation-support facilities, a target language can easily become a morass of disparate constructs whose overall complexity mitigates any benefits attendant with the provision of simulation support. Kiviat offers a compelling argument against such language extensions.[24] Certainly, when compared to the architectural harmony within which simulation support is implemented in SIMSCRIPT and Simula, most language "add-ons" seem poorly conceived. The disadvantage of language extension can be overcome by integration of HLA constructs within "pure" SPLs. However, HLA integration may be quite invasive to the language design, requiring change in fundamental areas. For example, the time flow mechanism must be modified to support the *request-grant* service cycle. Bélanger, Byers and Lam[25] describe a language extension and integration effort that involves the addition of HLA capabilities to the OSim framework.

The architectural extension approach has a similar advantage to the language integration approach in that it supports the synthesis of simulation and distributed systems CFs. However, this approach raises a question regarding the relationship between *architecture* and *language* in a simulation-support role. We consider this question below.

3.3. Architecture and Language Issues

According to Nance[26] a simulation programming language must provide:

1. Generation of random numbers, so as to represent the uncertainty associated with an inherently stochastic model.
2. Process transformers, to permit uniform random variates obtained through the generation of random numbers to be transformed to a variety of statistical distributions.
3. List processing facilities, so that objects can be created, deleted, and manipulated as sets or as members, added to and removed from sets.
4. Statistical analysis routines, to provide the descriptive summary of model behavior so as to permit comparison with system behavior for validation purposes and the experimental analysis for both understanding and improving system operation.
5. Report generation, to furnish an effective presentation of potentially large reams of data to assist in the decision making that initially stimulates the use of simulation.
6. Timing executive, or time flow mechanism, to provide an explicit representation of time.

[§]As an aside, it is arguable that some of the popular high-level languages today are actually *high-level* languages at all.

These facilities are considered minimal to support the simulation modeling task. Nance notes that every simulation programming language provides these components to some degree. The HLA provides support for list processing and time flow. Should support for the other elements be provided? The answer to this question isn't clear. Is there a *proper* separation of responsibilities between a simulation-support architecture and a simulation-support language?

It is reasonable to assert that an architecture should be as simple as possible, and that the primary burden for modeler support must reside with the simulation language. Is the HLA as simple as possible? Certainly to provide synchronized distributed execution, the architecture must provide time flow. The management of public objects mandates list processing capabilities. But what about random number generation and process transformation? While a few experiments have been conducted regarding the use of the HLA to support analysis, the majority of HLA applications (and DIS and ALSP applications) are geared toward simulation-based training. Many of the difficult statistical issues attendant with the application of simulation are not manifest in a training environment. However, it seems reasonable that the correlation of random number streams across distributed entities might be required in certain analytical settings. It may be possible to circumvent the absence of this facility through a properly coordinated use of identical random number generators and seed values. However, generation of random variates through the RTI might prevent rounding discrepancies or race conditions.

The SPL requirements Nance identifies are not *necessarily* requirements for a simulation-support architecture, but they must be viewed as *candidate* requirements and reasons for their exclusion should be explicitly noted.

4. CONCLUSIONS

The HLA has been mandated as a DoD standard. All DoD modeling and simulation (M&S) efforts must comply with the HLA, receive a waiver, or be retired by 2001.[2] The DoD has made similar proscriptions in the past – with mixed success. The recently rescinded Ada mandate is an example. Could HLA realize the same fate as a DoD standard that befell Ada? It is possible. On the other hand, the HLA is potentially quite suitable to provide a framework for distributed simulation systems interoperability within the broader simulation community. The cultural demand for interoperability may drive both the commercial and academic communities to define solutions to systems interoperation. Their requirement for interoperability may enable the adoption of HLA – even if it does not completely meet their needs. But it might not.

The HLA currently does not easily accommodate the approaches to simulation model conceptualization and development that are widely applied within the discrete event simulation community. Support for these traditional conceptual frameworks within the context of the HLA is quite possible, and mechanisms for this support have been described here. As part of MITRE's ongoing web-based simulation support research effort, we are integrating HLA within *Silk,*[27,28] a Java-based simulation-support language. The details of that effort will be reported separately.

REFERENCES

1. P. A. Fishwick, "Web-based simulation: Some personal observations," in *Proceedings of the 1996 Winter Simulation Conference*, J. M. Charnes, D. M. Morrice, D. T. Brunner, and J. J. Swain, eds., pp. 772–779, (Coronado, CA), 8-11 December 1996.
2. U.S. Department of Defense, DoD High Level Architecture (HLA) for Simulations, September 1996. Memorandum signed by USD(A&T).
3. E. H. Page, A. Buss, P. A. Fishwick, K. J. Healy, R. E. Nance, and R. J. Paul, "The modeling methodological impacts of web-based simulation," in *Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation*, P. A. Fishwick, D. R. C. Hill, and R. Smith, eds., pp. 123–128, (San Diego, CA), 11-14 January 1998.
4. J. S. Dahman, R. M. Fujimoto, and R. M. Weatherly, "The Department of Defense high level architecture," in *Proceedings of the 1997 Winter Simulation Conference*, S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, eds., pp. 142–149, (Atlanta, GA), 7-10 December 1997.
5. L. D. Voss, *A Revolution in Simulation: Distributed Interaction in the '90s and Beyond*, Pasha Publications, Arlington, VA, 1993.
6. E. H. Page, B. S. Canova, and J. A. Tufarolo, "A case study of verification, validation and accreditation for advanced distributed simulation," *ACM Transactions on Modeling and Computer Simulation* **7**(3), pp. 393–424, 1997.

7. R. M. Weatherly, A. L. Wilson, and S. P. Griffin, "ALSP – theory, experience, and future directions," in *Proceedings of the 1993 Winter Simulation Conference*, G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. E. Biles, eds., pp. 1068–1072, (Los Angeles, CA), 12-15 December 1993.
8. R. M. Weatherly, A. L. Wilson, B. S. Canova, E. H. Page, A. A. Zabek, and M. C. Fischer, "Advanced distributed simulation through the aggregate level simulation protocol," in *Proceedings of the 29th Hawaii International Conference on Systems Sciences*, vol. 1, pp. 407–415, (Wailea, HI), 3-6 January 1996.
9. U.S. Department of Defense, High Level Architecture Object Model Template, February 1998. Version 1.3.
10. U.S. Department of Defense, High Level Architecture Interface Specification, February 1998. Version 1.3.
11. U.S. Department of Defense, High Level Architecture Rules, February 1998. Version 1.3.
12. R. M. Fujimoto, "Zero lookahead and repeatability in the high level architecture," in *Proceedings of the 1997 Spring Simulation Interoperability Workshop*, (Orlando, FL), 3-7 March 1997.
13. E. J. Derrick, O. Balci, and R. E. Nance, "A comparison of selected conceptual frameworks for simulation modeling," in *Proceedings of the 1989 Winter Simulation Conference*, E. A. MacNair, K. J. Musselman, and P. Heidelberger, eds., pp. 711–718, (Washington, D.C.), 4-6 December 1989.
14. O. Balci, R. E. Nance, E. J. Derrick, E. H. Page, and J. L. Bishop, "Model generation issues in a simulation support environment," in *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R. P. Sadowski, and R. E. Nance, eds., pp. 257–263, (New Orleans, LA), 9-12 December 1990.
15. E. J. Derrick, *A Visual Simulation Support Environment Based on a Multifaceted Conceptual Framework*. PhD thesis, Virginia Tech, Blacksburg, VA, April 1992.
16. M. R. Lackner, "Digital simulation and system theory," Tech. Rep. SDC SP-1612, System Development Corporation, Santa Monica, CA, 1964.
17. P. J. Kiviat, "Digital computer simulation: Modeling concepts," Tech. Rep. RM-5378-PR, RAND Corporation, Santa Monica, CA, January 1967.
18. P. J. Kiviat, "Digital computer simulation: Computer programming languages," Tech. Rep. RM-5883-PR, RAND Corporation, Santa Monica, CA, January 1969.
19. G. S. fishman, *Concepts and Methods in Discrete Event Digital Simulation*, John Wiley and Sons, New York, NY, 1973.
20. R. E. Nance, "The time and state relationships in simulation modeling," *Communications of the ACM* **24**(4), pp. 173–179, 1981.
21. K. D. Tocher, "Review of simulation languages," *Operational Research Quarterly* **16**(2), pp. 189–217, 1965.
22. E. J. Derrick, "Conceptual frameworks for discrete-event simulation modeling," Master's thesis, Virginia Tech, Blacksburg, VA, August 1988.
23. U. Klein, S. Straßburger, and J. Beikrich, "Distributed simulation with JavaGPSS based on the high level architecture," in *Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation*, P. A. Fishwick, D. R. C. Hill, and R. Smith, eds., pp. 85–90, (San Diego, CA), 11-14 January 1998.
24. ACM Special Interest Group on Simulation, An Interview with Phil Kiviat, 1998. Part of *A Distinguished Lecture Series in Simulation,* to appear in *Simulation Digest* and online at `http://www.acm.org/sigsim/`.
25. J.-P. Bélanger, C. Byers, and L. Lam, "Osim framework: Experience of an in-flight refueling federation development using commercial tools," in *Proceedings of the 1997 Fall Simulation Interoperability Workshop*, pp. 667–676, (Orlando, FL), 8-12 September 1997.
26. R. E. Nance, "A history of discrete event simulation programming languages," in *Proceedings of the Second ACM SIGPLAN History of Programming Languages Conference*, (Cambridge, MA), 20-23 April 1993. Reprinted in *ACM SIGPLAN Notices,* **28**(3), pp. 149-175.
27. K. J. Healy and R. A. Kilgore, "Silk: A java-based process simulation language," in *Proceedings of the 1997 Winter Simulation Conference*, S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, eds., pp. 475–482, (Atlanta, GA), 7-10 December 1997.
28. R. Kilgore and K. Healy, "Java, enterprise simulation and the Silk simulation language," in *Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation*, P. A. Fishwick, D. R. C. Hill, and R. Smith, eds., pp. 195–200, (San Diego, CA), 11-14 January 1998.