The Role of Collaborative DEVS Modeler in Federation Development
Hessam S. Sarjoughian
Bernard P. Zeigler
Electrical & Computer Engineering Department
University of Arizona, Tucson, AZ 85721-0104
Email: {hessam|zeigler}@ece.arizona.edu
Web: http://www.acims.arizona.edu

Keywords: Collaborative Modeling, DEVS, Distributed Simulation, FEDEP, HLA, OMT, RTI, System Entity Structure.

ABSTRACT: Well-defined, systematic distributed modeling paradigms are an imminent necessity. Such paradigms are paramount to the success of DoD’s increasing reliance on simulation. Although a well-defined federation development and execution process, FEDEP, has been defined to support HLA-compliant simulations, it does not, by itself, deal with the issue of collaboration among all participants in such activities. To address this critical need, modeling and simulation methodologies should be examined in terms of how well they take into account the problems inherent in collaborative work. Given such problems, it is impractical to rely on conventional means (e.g., face-to-face meetings and generic collaborative tools) to “hook up” modelers with one-another from different organizations to create models across space and time constraints. In this paper, we describe the Collaborative DEVS Modeler (CDM) environment which is based on the DEVS modeling framework. Using CDM, subject matter experts and simulation modeling professionals can develop modular, hierarchical models within its virtual collaborative modeling workspace. We discuss CDM’s architecture and functionalities from two point of views: as a generic Computer Supported Collaborative Work (CSCW) tool and as specific support environment for distributed modeling, a form of work that has its own particular characteristics. Within this context, distributed modeling concepts and their implications for constructing successful distributed simulations are presented. Although, CDM works within the DEVS framework, it is applicable much more generally to the collaborative development of HLA-compliant simulations. We describe the complementary features of the DEVS Modeling concepts with those of the HLA/Object Model Template. Specifically, we discuss how CDM modeling artifacts support the conceptual model development and federation design phases of the HLA FEDEP model development with the added benefit of working within a sound theory-based collaborative modeling framework.

1. INTRODUCTION
The High Level Architecture (HLA) objective is to enable simulation interoperability and reuse, thus enabling various simulation types (logical and real) to interoperate. The HLA standard is a modeling and simulation interface specification and to a lesser extent an specification for the design space of HLA-compliant simulations. The HLA/OMT specification, one of the three pieces of the HLA standard, defines the structure of HLA object models. The other two define (1) federate interface specification and (2) rules [1]. The interface specification is mainly concerned with management of an HLA federation without regard to the details/internals of the federates. For example, the interface must enable a federate to join or leave a federation. HLA rules apply to OMT’s simulation and federation object models. Five rules govern the federations declarations and execution given the HLA Run Time Infrastructure (RTI) while complying with the federate interface specification. The remaining five rules specify what federates may and may not do.

The Discrete-event System Specification (DEVS) is a system-theoretic based approach to modeling of discrete-event systems and a large class of continuous systems [2]. The DEVS modeling formalism enable characterization of systems in terms of hierarchical modules with well-defined interfaces. Due to its system-theoretic foundations, DEVS modeling paradigm naturally maps into object-orientation implementation and consequently has been implemented in sequential, parallel, and distributed environments [3]. In addition to DEVS Modeling constructs, the System Entity Structure (SES) in conjunction with rule-based system have been proposed and employed to deal with design choices [4, 5]. The combination of DEVS and SES provides a suite of modeling capabilities for full-fledged

---

1 This work was supported in part by Advance Simulation Technology Thrust (ASTT) DARPA Contract N61339-97-K-0007.
dynamical systems containing intelligent features (e.g., rule-based, fuzzy logic, and neural networks.)

The Collaborative DEVS Modeler (CDM) allows geographically dispersed modelers (users) to collectively compose modular, hierarchical models [6]. Dispersed subject matter experts and modeling professionals alike can collaborate both synchronously and asynchronously via shared diagrams, BBS style messaging, and a text-based chat system. CDM enforces model synthesis that adhere to the DEVS modeling paradigm [2]. In the following sections, we discuss modeling capabilities offered by HLA/OMT, DEVS, and SES. We further identify some preliminary relationships among these complementary and to some extent alternative modeling approaches.

2. HLA FEDEP

In order to support the HLA community to build HLA-compliant simulations, the HLA Federation Development and Execution Process (FEDEP) model comprised of five steps has been proposed (see Figure 1). The HLA FEDEP promotes an “iterative waterfall software process” to support development of HLA-compliant simulations [7]. This model is similar to software development processes in which a series of five steps are followed to produce a software system which satisfies a client’s objectives and requirements [8]. Given the five steps, steps 2 (develop federation conceptual model) and 3 (design and develop federation) are essential in representing the federation and simulation object models.

Knowledge such as the data to be shared among federates as well as intrinsic capabilities of each simulation represent the model of an HLA-compliant distributed simulation. These outcomes of these two steps in conjunction with the HLA rules is stated to ensure simulation interoperability while supporting reuse [7].

Examination of FEDEP methodology reveals that detailed sub-methodologies are necessary to support the development of HLA-compliant simulation. In particular, steps 2 and 3 of the FEDEP need to be supported with (1) functional (dynamical) modeling primitives and (2) alternative modes of operation for teamwork. The FEDEP model is neutral toward what kinds of development modes (e.g., synchronous collaboration) may be employed in developing object models and other HLA related models.

In this writing, we propose the collaborative DEVS modeling paradigm to support conceptual model development and federation design. Not only CDM (a collaborative DEVS modeling environment) can support FOM and SOMs interface specification, it can also support model construction and synthesis at the functional level. The collaborative DEVS modeling is the amalgamation of DEVS modeling principles and the Computer Supported Collaborative Work (CSCW) [9]. As shown in Figure 1, the FEDEP steps can potentially be carried out in Conventional Meetings (CM), Synchronous Collaboration (SC), and Asynchronous Collaboration (AC) modes.
2.1. HLA/OMT

The concept of an HLA Object Model Template (OMT) plays a central role in building HLA-compliant simulation as purported by HLA standard. The primary role of HLA/OMT is to document HLA-relevant information about federates and federations [10]. The HLA/OMT specifies two object models: Federation Object Model (FOM) and Simulation Object Model (SOM). Specifically, there are two main technical objectives for the HLA/OMT 1.3 Specification [10]. The first objective is aimed at FOM and thus it is to provide a common specification for exchange of data and coordination among members of a federation. Correspondingly, the second objective is aimed at SOMs. It is to provide a common, standardized mechanism for describing the capabilities of potential federation members. To achieve these objectives, the HLA/OMT is devised as a collection of three components: (1) object classes, (2) interaction classes, and (3) routing space and their dimensions. (More precisely, the HLA/OMT is divided into seven components: object model identification, object class structure, interaction class structure, attributes, parameters, routing space, and FOM/SOM lexicon.) Object and interaction classes represent the data that can be exchanged among federates. Object classes contain attributes and interaction classes contain parameters. Each object class has a lifetime associated with it in contrast to an interaction class (event) which can only exist instantaneously. The remaining element of the HLA/OMT specification is to facilitate efficient data distribution (routing) among federates.

HLA objects can belong to Federation Object and Simulation Object Models. An FOM deals with the issues surrounding the decomposition of a federation into its federates. In particular, it specifies a federation’s object and interaction classes along with their attributes and parameters which collectively establish a necessary but insufficient contract supporting interoperability among federates. The FOM can also optionally contain supplementary modeling constructs. Unlike FOM, a SOM deals with intrinsic capabilities of federates such as their dynamic behavior (operations) to the extent of capturing interactions, publications, and subscriptions for a federate.

We can view modeling concerns/issues contained in the HLA/OMT (FOM and SOM) in terms of federation rules and interface specifications [11]. The HLA standard rules are divided into federation and federate rules where each set of rules specify the constraint under which federates and federations and interface specification work with one another. The federation rules are mainly concerned with execution and therefore with HLA RTI. The modeling aspect of these rules are tied to the FOM and SOM where in
a federation execution, all data exchanges must be in accordance with FOM data, and each SOM can be owned by at most one federate.

We can examine HLA rules 7 through 9 which correspond to the federates. Rule 7 states that federate shall be able to (a) update and/or reflect attributes and (b) send and/or receive interactions. Dynamic transfer and acceptance of attribute ownership is captured via Rule 8. Rule 9 addresses dynamic conditions (e.g., threshold setting) under which attributes are updated. Thus, these rules capture chiefly interface modeling constructs that are essential in modular model representation and the constraints by which models interoperate. Note that these rules do not take into account the internals of federates participating in a federation. The specifics of what might comprise a federate’s behavior are unspecified by HLA rules and OMT. As mentioned earlier, such knowledge is considered supplementary as far as building HLA-compliant simulations. Consequently, the interactions among the components of a modular, hierarchical model (federate) neither can be captured by OMT nor do they have to comply with any of HLA rules. The exclusion of each federate’s internals is due to the fact that a federate can be anything represented by HLA object models.

The specification of federation object and simulation object models can be carried out by a single modeler or a group of modelers or subject matter experts. In the case of a single modeler, either existing FOMs/SOMs are adapted or new ones are developed from following requirements definition phase of the FEDEP. Of course, given the vision of HLA standard, federation and simulation object models are more likely to be developed in a group setting using CSCW-based environments such as CDM.

3. SYSTEM ENTITY STRUCTURE
A System Entity Structure (SES) provides the means to represent a family of models as a labeled tree [4, 12]. Two of its key features are support for decomposition and specialization. The former allows representing a large system in terms of smaller systems. The latter supports representation of alternative choices. Specialization enables representing a generic model (e.g., a computer display model) and its specialized variations (a flat panel display or a CRT display.) Based on SES axiomatic specifications, a family of models can be represented and pruned to study and experiment with design choices (alternatives.) An important, salient feature of SES is its ability to represent models not only in terms of their decomposition and specialization, but also aspects (SES represents alternative decompositions via aspects.) With respect to Figure 2, modeling services of DEVS and SES primarily match up with the Modeling layer and the model design space matches up with Search/Decision layers.

Figure 2: Architecture for Modeling & Simulation

4. DISCRETE EVENT SYSTEM SPECIFICATION
The Discrete Event System Specification (DEVS) [2] modeling approach supports capturing a system’s structure in terms of atomic and/or coupled models where coupled models are hierarchical satisfying closure under coupling [5]. While a part of a system can be represented as an atomic model with well-defined input/output interfaces, a system represented as a DEVS coupled model designates how systems can be coupled together to form system of systems. Such coupled models also have the same input/output interfaces. Given atomic models, DEVS coupled models can be formed in a straightforward manner. Both atomic and coupled models can be simulated using sequential and/or various forms of parallel computational techniques [13, 14]. A DEVS atomic model represents a component’s dynamic behavior and has a well-defined mathematical structure as follows:

\[
\text{Atomic Model} = \langle X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta \rangle
\]

where

- \(X\) set of input events,
- \(S\) set of sequential states,
- \(Y\) set of output events,
- \(\delta_{\text{int}}\) internal transition functions specifying state transitions due to internal events,
- \(\delta_{\text{ext}}\) external transition functions specifying state transitions due to external events,
- \(\lambda\) output function generating external events as outputs,
- \(ta\) time advance function.

\[\text{2 OMT is a constrained version of object-orientation since HLA objects primarily contain data and no methods.}\]
and simulation layers, respectively. The simulation and HLA/RTI are matched roughly to the modeling simulation [15, 17, 18]. In this environment, DEVS layers have been devised to support HLA-compliant DEVS/HLA-C++, the simulation and modeling collectively support the enterprise of M&S and its domains, providing support for a system’s life-cycle (such as analysis, design, execution). The inherent multidisciplinary characteristics of many existing and future systems, however, necessitated distributed simulation environments such as DEVS/HLA [15]. As suggested in [16], a M&S architecture (see Figure 2) provides a comprehensive M&S framework. This architecture consists of six layers where lower layers provide services for the upper layers: (1) Network (the lowest layer), (2) middleware, (3) simulation, (4) modeling, (5) search/decision, and (6) collaboration.

In the architecture depicted in Figure 2, all layers collectively support the enterprise of M&S and its variations such as Simulation Based Acquisition. In DEVS/HLA-C++, the simulation and modeling layers have been devised to support HLA-compliant simulation [15, 17, 18]. In this environment, DEVS and HLA/RTI are matched roughly to the modeling and simulation layers, respectively. The simulation layer uses the DEVS distributed simulation engine in addition to the RTI [19].

Rows of Table 1 [20] illustrate how various entities from the HLA/OMT correspond to those of DEVS/SES. For example, loosely speaking, both SOM and AM represent dynamic behavior, while FOM and CM represent decomposition of a federation (coupled model) in terms of its federates (components.) The first and second row show that both HLA/OMT and DEVS can used to model (represent) FOMs and SOMs. A federation and its members can also be represented in SES without the ability to specify federates interconnectivities. The fourth row of the table indicates that the routing knowledge represented in the HLA/OMT correspond neither to DEVS nor SES. However, considering that interoperation among federates are in part captured via subscription and updating regions (routing space), then coupled models essentially are representing routing space, but from a higher level of abstraction. That is, coupling of models in DEVS does not consider data distribution management and the techniques that are employed by RTI to reduce data delivery to federates. The DEVS coupling, however, is more generalized compared to OMT Routing since it supports hierarchical interconnections. The System Entity Structure might also be extended to support routing, although such a hypothesis has yet to be shown valid. Note that DEVS coupled model can be either a federation or federate since based on the HLA standard, a federation, in essence, is a single-layer hierarchy.

Given DEVS, HLA/OMT, and SES, we can begin an examination of their modeling features with respect to one another (see Table 1.) Our examination is from two points of view: Interface Modeling (HLA/OMT) and Dynamical Modeling (DEVS/SES.) Interface Modeling is defined as modeling of composite systems where its dynamics are either unspecified or at most specified at the input/output level (i.e., input and output values are identified.) Dynamical Modeling can subsume interface modeling by not only supporting interface specification but also dynamics of the composite systems via specification of the internal dynamics of its sub-systems. For example, DEVS/SES modeling primitives and constructs support interface modeling at the level of first-class objects that have full representation and functionality. Unlike DEVS/SES, HLA/OMT specifies quasi-objects (i.e., object and interaction classes) which only contain attributes and parameters.
### Table 1: Complementary HLA/OMT & DEVS/SES Modeling Constructs

<table>
<thead>
<tr>
<th></th>
<th>HLA/OMT</th>
<th>DEVS</th>
<th>System Entity Structure (SES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM</td>
<td>Atomic Model</td>
<td>N/A</td>
<td>Decomposition</td>
</tr>
<tr>
<td>FOM</td>
<td>Coupled Model</td>
<td>specialization &amp; aspect</td>
<td></td>
</tr>
<tr>
<td>Routing Space</td>
<td>partially supported</td>
<td>not supported</td>
<td></td>
</tr>
</tbody>
</table>

The interpretation of Table 1 is as follows. We begin with two sets of observations:

1.1 HLA/OMT characterizes SOM, FOM, and Routing Space;
1.2 DEVS characterizes Atomic and Coupled Models;
1.3 SES supports decomposition, specialization, and aspect.

2.1 SOM and Atomic Model constructs are equivalent at the level of representing a component’s state space;
2.2 FOM, DEVS Coupled Model, and SES decomposition can represent system of systems (decomposition of a parent model in terms of its children) via their couplings.

Given the above observations, SES does not support characterization (modeling) of a component’s dynamics (see row 1). SES, DEVS, and FOM can represent model decomposition (see row 2). Neither HLA/OMT nor DEVS modeling constructs are able to support specialization and aspects (see row 3). While DEVS/HLA is able to support a form of routing space (i.e., predictive filtering [21]), SES provides no support for components (federates or atomic models) to efficiently communicate with one another (see row 4). A suite of data management approaches including quantization and spatial encounter prediction are discussed in Joint Measure which is a DEVS/HLA compliant simulation environment for mission effectiveness [21].

The claim that HLA/OMT supports neither specialization nor aspect features as supported by SES requires justification. Our reasoning is based on the belief that while HLA standard asserts that SOMs can be written in such a way that they can be used in alternative federations, there exists no systematic support for representing design spaces (family of models.) In the HLA five-step development process, FOMs are purported to be reusable as well during the design phase either through their modifications or their individual pieces. Again, a framework to carry out reusability with some well-defined scheme (syntax and semantics) is not provided by HLA standard. The consequence is that such knowledge cannot readily lend itself to automation where it is most desirable.

Therefore, based on the “modeling” features offered by HLA/OMT and DEVS/SES, we can view HLA/OMT modeling constructs as abstractions (higher-level) when compared with those offered by DEVS/SES. That is, from the standpoint of object-oriented modeling DEVS/SES is a superset of HLA/OMT with the exception of the routing mechanism. The higher level HLA/OMT modeling constructs are due to choices that have been made in the HLA standard and HLA/OMT in particular to accommodate non object-oriented models. The HLA standard is intended to support a lower common denominator through which declare knowledge about a federation (e.g., FOM vocabulary) while allowing details of models to be encapsulated inside respective SOMs.
One difference between HLA/OMT and DEVS/SES is that while the former allows abstract modeling (federate/federation interfaces — interaction and object classes), the latter provides a comprehensive set of modeling constructs. Stated differently, due to system theoretic basis of DEVS/SES, the FOM and SOM can be captured in a straightforward/formal setting. The object and interaction classes can be derived from DEVS atomic and coupled components (see Table 2.) Any subset $S'$ of an atomic component state set $S$ can represent the attributes of an object class. Attributes associated with a Coupled Model ($S_{CM}$) can be mapped into Object Class Hierarchy ($S_{OCH}$). Since the state set of a coupled model is a union of state sets of its components states (cross product), the attributes of an object class hierarchy is subset of the coupled model state set.

$$S_{CM} \supseteq S_{OCH}$$

where $S_{CM} = S_{AM} \cup S'_{CM}$

$i = 0, \ldots, m, j = 0, \ldots, n$, and $i \neq j$.

The highest level coupled model state set ($S_{CM}$) can have atomic model state sets ($S_{AM}$) and coupled model state sets ($S'_{CM}$). Note that since coupled model state set is a superset of the attributes of an object class hierarchy, $S_{CM} \rightarrow S_{OCH}$ is not unique.

Conjecture: hierarchical relationships among the state sets of a coupled model components can be mapped to a finite number of object class hierarchies where any one of these object class hierarchies preserves the DEVS class hierarchy.

Corollary: Due to the object-orientation of the DEVS models and access modifiers associated with their state variables (e.g., static and private), object class hierarchies which do not preserve DEVS object hierarchies require new/modified DEVS models.

Input/output values which are derived from any DEVS atomic component state set can be represent parameters and therefore an interaction class. Note that in DEVS, input/output values made available at input/output ports are ultimately derived from the atomic model’s state set and constants. Similar to object class hierarchies, interaction class hierarchies can also be derived from hierarchical DEVS models.

Note that DEVS naturally support object class and interaction class hierarchies due to its support for object-oriented, hierarchical model construction. The consequence is that object and interaction class hierarchies derived from DEVS have a sound basis due to (1) the theoretical underpinning of DEVS and (2) its realization in a fully object-oriented language. In contrast, in HLA/OMT object and interaction object models may not map one-to-one to the internals of a non-object-oriented object and how its federates might relate to one another. DEVS/HLA-C++, an HLA-Compliant simulation environment utilizes both object and interaction classes [17]. However, in one respect, the HLA standard provides distributed modeling constructs (i.e., data distribution management) that are not accounted for by the traditional DEVS/SES. Recent extensions to DEVS (DEVS/HLA), however, has introduced “distance based quantization” which supports efficient data transfer among federates [15]. We also observe that DEVS modeling constructs are a superset of those offered by finite state machines (FSM). Therefore the discussion of the HLA/OMT modeling issues with respect to the DEVS/SES is applicable to FSM and its equivalents.

Figure 3 illustrates three alternative federation assignments as one federate consisting of all three components (a), two federates with a two to one split (b), and three federates (c). When DEVS components within a federate are assigned to their own federates (Figure 3 (d, e)), their message exchanges are handled by RTI as interactions. DEVS/HLA provides an implementation of this approach. For example, DEVS internal coupling from fireOut to fireIn can be mapped into the fire interaction. In DEVS/HLA interaction parameters are represented as instances of the HLAport class which are exchanged across the network. The existing DEVS ports fireOut and fireIn are then coupled with the interaction port fire as shown in Figure 3 (e).

6. COLLABORATIVE DEVS MODELER

We can distinguish two types of modeling activities: model construction and model synthesis [16]. Model construction mostly deals with identifying dynamics of models while model synthesis concerned with synthesizing coupled models (federates and federations) given existing atomic and/or coupled models (federates). We may also view SOMs as the outcome of the model construction activity and FOMs as the product of the model synthesis activity. Unfortunately, the boundary between model construction and synthesis is rather imprecise due to nature of modeling.

The collaborative DEVS modeler is based on the integration of two disciplines: Modeling and Distributed Object Computing [6]. The Collaborative DEVS Modeler approach to modeling is based on the concept of a session which is a “loosely bounded
workspace” within which a group of knowledge modelers develop a model as a team. This conceptual view of CDM illustrates two basic issues: distribution of modelers (knowledge workers) and their resources across time and space.

The Collaborative DEVS Modeler can support synchronous and asynchronous synthesis of federations. Furthermore, CDM can be used to support construction of individual federates where each federate requires collaboration among multiple users who may or may not be geographically distributed. Synchronous collaboration can be supported by complementary capabilities such as text-based and video-teleconferencing tools in order to support rich collaboration among modelers. These capabilities can be tailored for modeling by providing representations of common modeling primitives and enforcing correct modeling constructs. Asynchronous collaboration also benefits from the availability of modeling primitives and semantics enforcement. Due to the nature of modeling, it is further necessary that models be persistent and retrievable by modelers anywhere on a network.

Figure 4 depicts CDM’s representation of an AircraftModel model in terms of its atomic (Controls, Cabin) and coupled (Navigation) components. The modularity principle supported by the DEVS modeling paradigm enables a modeler to develop her own models (i.e., Navigation, Controls, and Cabin) largely independent of how her collaborator develop the EnvironmentalModel and its components (e.g., Terrain, telemetryDevice, and Sensor.) Using CDM, a modeler would be able to individually devise the components of the AircraftModel and their interoperation via output/input connections while making the models available to her collaborator as an encapsulated model with its own input and output ports.

![Figure 3: Alternative Federation Compositions (a, b, c); Mapping DEVS Couplings into HLA Interactions (d, e)](image-url)
Figure 4: Component Representation of the Aircraft Model Comprised of Atomic and Coupled Models

Figure 5: Collaborative Modeling & Distributed Simulation Environment Supporting Development of Object Models and Federation Configuration and Execution

Figure 5 depicts two simulation federations (LMMS/UA and YPG/UA) with federations representing Lockheed Martin Missiles Space (LMMS), Yuma Proving Ground (YPG), and University of Arizona (UA). These federate represent development of ongoing HLA-compliant simulations efforts underway among University of Arizona, Lockheed Martin Missiles Space, and Yuma Proving.
Ground. The collaborative DEVS modeling tool (CDM) can enable the modelers/simulationists from UA, LMMS, and YPG to collaborate with one another to develop federates and federations. The arrows pointing from Simulation Federation LMSS/UA and YPG/UA to HLA/OMT and DEVS/SES illustrate interoperability of DEVS/HLA-C++ and CDM. This diagram depicts how modeling, simulation, and collaboration layers can be brought together to support basic collaborative modeling and simulation activities such as configuration, execution, control, and monitoring. It is suggested that mode of operations for Configuration & Execution and Control & Monitoring to be synchronous and asynchronous, respectively.

The data (attributes and parameters) to be exchanged among federates would be captured during building of atomic and coupled models. Atomic models would contain instances from the OMT object and interaction classes. The CDM tool can be extended to make federation and simulation object models available in tabular form as it is mandated by HLA standard. The generation of such knowledge is straightforward since the CDM models are based on components with encapsulated dynamics and input/output interfaces. Modelers at the three collaborating sites can develop federate models and configure federation models which can be simulated using DEVS-Compliant simulation environments such as DEVS/HLA [23]. Using an extended CDM tool, federates/federation can then be started, suspended, and stopped. Therefore, to overcome problems inherent in collaborative model development and execution, CDM can offers a theory-based tool supporting collaborative model development and federation design stages.

7. RELATED AND FUTURE WORK
From the general perspective of support for collaborative modeling, there exist a variety of tools. Since 1990s, Computer Supported Collaborative Work (CSCW) research has focused [9, 24] on supporting collaboration in many applications including modeling as prevalent in software engineering. These environments are generally devised for the sake of creating models that are not expected to be executed in a full-fledged distributed simulation environment. In cases where models can be simulated in this manner, support for the modeling task is ad hoc. Furthermore general CSWS systems (e.g., [25]) cannot support verification and validation of models. Other tools such as Distributed Simulation Exercise Construction Toolset (DiSECT) are expected to support a variety of capabilities such as exercise planning, control, monitoring, and analysis. While at the time of this writing we have access to neither to documentation nor the tool itself, its capabilities appear to differ from what has been discussed in this paper. In particular, CDM focus is on DEVS model development and their transformation into federation and simulation object models.

In terms of future research, it would be valuable to have a suitable characterization of development modes (e.g., synchronous collaboration) and their use in practical settings. Automated mapping and generation of FOMs and SOMs from models developed using the CDM environment would support providing OMT documentation required by any HLA-compliant simulation. Furthermore, support for model design space, reuse, and management using SES concepts can enable not only systematic DEVS model development, but also FOM and SOM specification.

In conclusion, while HLA standard provides the modeling and simulation community with concepts and tools to build distributed simulations, the HLA/OMT specification provides partial modeling constructs (interface modeling.) The DEVS/SES provides added modeling concepts and constructs (dynamics and composition alternatives modeling) that are outside of the scope of HLA/OMT framework. We are continuing with our study of modeling concepts (modular, hierarchical, and interoperability) with the objective to rigorously characterize the trio of HLA/OMT, DEVS, and SES.

ACKNOWLEDGEMENT
This paper was reviewed by Gary Bundy. We appreciate his constructive review.

REFERENCES


HESSAM S. SARJOUGHIAN is Assistant Research Professor of Electrical and Computer Engineering at the University of Arizona. His present research interests are in theory, methodology, and practice of collaborative/distributed & simulation. Currently, he is leading the design and development of CDM to support distributed simulation.

BERNARD P. ZEIGLER is Professor or Electrical and Computer Engineering at the University of Arizona, Tucson. He has written several foundational books on modeling and simulation theory and methodology. He is currently leading a DARPA sponsored project on DEVS framework for HLA and predictive contracts. He is a Fellow of the IEEE.