Open access • Proceedings Article • DOI:10.1109/ICSMC.2005.1571646

Enhancing DoDAF with a DEVS-based system lifecycle development process

Bernard P. Zeigler, Saurabh Mittal

Institutions: University of Arizona

Published on: 10 Oct 2005 - Systems, Man and Cybernetics

Topics: Operational View, TAFIM, DEVS, System lifecycle and Software architecture

Enhancing DoDAF with a DEVS-Based System Lifecycle Development Process

Bernard P. Zeigler
Arizona Center of Modeling & Simulation
ECE Department
University of Arizona, Tucson
zeigler@ece.arizona.edu

Saurabh Mittal
Arizona Center for Modeling & Simulation
ECE Department
University of Arizona, Tucson.
saurabh@ece.arizona.edu

Abstract *- A recent DoD mandate requires that the DoD Architectural Framework (DoDAF) be adopted to express high level system and operational requirements and architectures. DoDAF is the basis for integrated architectures and provides broad levels of specification related to operational, system, and technical views. The combination of DoDAF Operational views, which capture the requirements of the architecture, and System views which provide its technical attributes, forms the basis for semi-automated construction of the needed simulation models. In this paper, we describe an approach to support specification of DoDAF architectures within a development environment based on DEVS (Discrete Event System Specification). The result is an enhanced system lifecycle development process that includes both development and testing in an integral manner. This paper discusses the motivation to carve out a methodology to develop DEVS models for any DoDAF-UML architecture specifications and empower DoDAF with integrated M&S.*

Keywords: DoDAF, Simulation-based design, DEVS. Bifurcated Development Process

1 Introduction

A recent DoD mandate requires that the DoD Architectural Framework (DoDAF) be adopted to express high level system and operational requirements and architectures [1]. DoDAF is the basis for the integrated architectures mandated in DOD Instruction 5000.2 [2] and provides broad levels of specification related to operational, system, and technical views. Integrated architectures are the foundation for interoperability in the Joint Capabilities Integration and Development System (JCIDS) prescribed in CJCSI 3170.01D and further described in CJCSI 6212.01C [3,4]. DoDAF and other DoD mandates pose significant challenges to the DoD system and operational architecture development and testing communities since DoDAF specifications must be evaluated to see if they meet requirements and objectives, yet they are not expressed in a form that is amenable to such evaluation. However, DoDAF-compliant system and operational architectures do have the necessary information to construct high-fidelity simulations. Such simulations become, in effect, the executable architectures referred to in the DODAF document. However, DoDAF has completely overlooked the translation of DODAF-compliant architectures into models that are of sufficient fidelity to support architectural evaluation in capable simulation environments and does not dictate any specific M&S technology. Operational views capture the requirements of the architecture being evaluated and System views provide its technical attributes. Together these views form the basis for semi-automated construction of the needed simulation models.

DoDAF is a framework prescribing high level design artifacts, but leaves open the form in which the views are expressed. A large number of representational languages are candidates for such expression. For example, the Unified Modeling Language, (UML) and Colored Petri Nets (CPN) are widely employed in software development and in systems engineering. Each popular representation has strengths that support specific kinds of objectives and cater to its user community needs. By going to a higher level of abstraction, DoDAF seeks to overcome the plethora of "stove-piped" design models that have emerged. Integration of such legacy models is necessary for two reasons. One is that, as systems, families of systems, and systems-of-systems become more broad and heterogeneous in their capabilities, the problems of integrating design models developed in languages with different syntax and semantics has become a serious bottleneck to progress. The second is that another recent DoD mandate also intended to break down this "stove-piped" culture requires the adoption of the Service Oriented Architecture (SOA) paradigm as supported in the development of Network Centric Enterprise Services (NCES) [5]. Under DoD direction, several contractors have begun to design and implement the NCES to support this strategy on Global Information Grid, a high-speed, high-capacity data network implemented on optical fiber technology. The result is that system development and testing must align with this mandate – requiring that all systems interoperate in a net-centric environment – a goal that can best be done by having the design languages be subsumed within a more abstract framework that can offer common concepts to relate to. However, as stated before, DoDAF does not provide a formal algorithmically-enabled process to support such integration at a detailed level.

2 DoDAF-to-DEVS mapping

We discuss a mapping of DoDAF architectures into a computational environment that incorporates dynamical systems theory and a modeling and simulation (M&S) framework. The methodology will support complex information systems specification and evaluation using advanced simulation capabilities. Specifically, the Discrete Event System Specification (DEVS) formalism will provide the basis for the computational environment with the systems theory and M&S attributes necessary for design modeling and evaluation.

We seek to employ the DoDAF-to-DEVS mapping to unify multiple model representations by expressing their high-level features within DoDAF and their detailed features as sub-classes of DEVS specifications. DEVS has been shown to be a universal embedding formalism, in the sense of being able to express any sub-class of discrete event systems, such as Petri Nets, Cellular Automata, and Generalized Markov Chains [6]. DEVS has also been employed to express a wide variety of more restricted formalisms, such as state machines, workflow systems, fuzzy logics, and others [7]. Moreover, DEVS environments have a long history of development and are now seeing ever increasing use in the simulation-based design of commercial and military systems [8]. Providing a DoDAF "front end" to a "back end" DEVS environment, will appeal to military information system designers facing the DoDAF and NCES mandates. Such designers will be able to retain their skills with representations familiar to them, while complying with DoDAF abstractions. At the same time they can see the results of their specifications evaluated via simulation-based execution of the model architecture. Moreover, since all mappings are into subclasses of DEVS, the resulting models can be coupled together and therefore can interoperate at the systems dynamics level. Thus this approach to the synthesis of system design formalisms leverages design and execution methodologies that are already used, or mandated for use, in commercial and military applications.

DEVS environments, such as DEVSJAVA, DEVS.C++, and others [9] are embedded in object-oriented implementations, thus supporting the goal of representing executable model architectures in an object-oriented representational language. As a mathematical formalism, DEVS, is platform independent, and its implementations adhere to the DEVS protocol so that DEVS models easily translate from one form (e.g., C++) to another (e.g., Java) [10] Moreover, DEVS environments, such as DEVSJAVA, execute on commercial-off-the-shelf desktops or workstations and employ state-of-the-art libraries to produce graphical output that complies with industry and international standards. DEVS environments are typically open architectures that have been extended to execute on various middleware such as DoD's HLA standard, CORBA, SOAP, and others [11,12.13.14]. Therefore, the proposed design architecture supports interfaces to other engineering and simulation and modeling tools – an example of such networking is provided by Lockheed's satellite cluster mission effectiveness simulator [15]. Furthermore, DEVS operation over a web-middleware (SOAP) enables it to fully participate in the net-centric environment of the NCES. As a result of recent advances, DEVS can support model continuity through a simulation-based development and testing life-cycle [16]. This means that the mapping of high-level DoDAF specifications into lower-level DEVS formalizations would enable such specifications to be thoroughly tested in virtual simulation environments before being easily and consistently transitions to operate in real environment for further testing and fielding.

3 DEVS System Specifications

In this section, we review some of the background required for discussion DEVS support of DODAF.

3.1 Hierarchy of Systems specifications

Systems theory deals with a hierarchy of system specifications which defines levels at which a system may be known or specified. Table 1 shows this Hierarchy of System Specifications (in simplified form, see [3]).

Level	Name	What we specify at this level
4	Coupled Systems	System built up by several component systems which are coupled together
3	I/O System	System with state and state transitions to generate the behavior
2	I/O Function	Collection of input/output pairs constituting the allowed behavior partitioned according to the initial state the system is in when the input is applied
1	I/O Behavior	Collection of input/output pairs constituting the allowed behavior of the system from an external Black Box view
0	I/O Frame	Input and output variables and ports together with allowed values

Table 1: Hierarchy of System Specifications

- At level 0 we deal with the input and output interface of a system.
- At level 1 we deal with purely observational recordings of the behavior of a system. This is an I/O relation which consists of a set of pairs of input behaviors and associated output behaviors.
- At level 2 we have knowledge of the initial state when the input is applied. This allows partitioning the input/output pairs of level 1 into non-

overlapping subsets, each subset associated with a different starting state.

- At level 3 the system is described by state space and state transition functions. The transition function describes the state-to-state transitions caused by the inputs and the outputs generated thereupon.
- At level 4 a system is specified by a set of components and a coupling structure. The components are systems on their own with their own state set and state transition functions. A coupling structure defines how those interact. A property of coupled system which is called "closure under coupling" guarantees that a coupled system at level 3 itself specifies a system. This property allows hierarchical construction of systems, i.e., that coupled systems can be used as components in larger coupled systems.

As we shall see in a moment, the system specification hierarchy provides a mathematical underpinning to define a framework for modeling and simulation. Each of the entities (e.g., real world, model, simulation, and experimental frame) will be described as a system known or specified at some level of specification. The essence of modeling and simulation lies in establishing relations between pairs of system descriptions. These relations pertain to the validity of a system description at one level of specification relative to another system description at a different (higher, lower, or equal) level of specification.

Based on the arrangement of system levels as shown in Table 1, we distinguish between vertical and horizontal relations. A vertical relation is called an association mapping and takes a system at one level of specification and generates its counterpart at another level of specification. The downward motion in the structure-to-behavior direction, formally represents the process by which the behavior of a model is generated. This is relevant in simulation and testing when the model generates the behavior which then can be compared with the desired behavior.

The opposite upward mapping relates a system description at a lower level with one at a higher level of specification. While the downward association of specifications is straightforward, the upward association is much less so. This is because in the upward direction information is introduced while in the downward direction information is reduced. Many structures exhibit the same behavior and recovering a unique structure from a given behavior is not possible. The upward direction, however, is fundamental in the design process where a structure (system at level 3) has to be found which is capable to generate the desired behavior (system at Level 1).

3.2 Framework for Modeling & Simulation

The *Framework for M&S* as described in [6], establishes *entities* and their *relationships* that are central to the M&S enterprise (see Figure 1). The entities of the framework are *source system, experimental frame, model,* and *simulator;* they are linked by the *modeling* and the *simulation* relationships. Each entity is formally characterized as a system at an appropriate level of specification within a generic dynamic system. See [6] for detailed discussion.

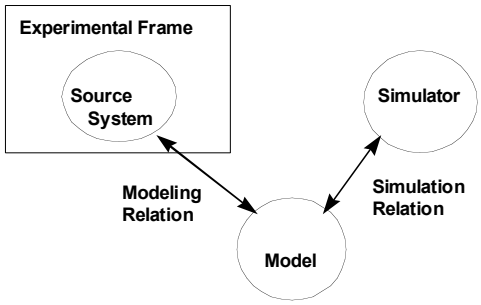

Figure 1. Framework Entities and Relationships

3.3 Model Continuity

Model continuity refers to the ability to transition as much as possible of a model specification through the stages of a development process. This is opposite to the discontinuity problem where artifacts of different design stages are disjointed and thus cannot be effectively consumed by each other. This discontinuity between the artifacts of different design stages is a common deficiency of most design methods and results in inherent inconsistency among analysis, design, test, and implementation artifacts [16]. Model continuity allows component models of a distributed real-time system to be tested incrementally, and then deployed to a distributed environment for execution. It supports a design and test process having 4 steps (see [16]),

1) Conventional simulation to analyze the system under test within a model of the environment linked by abstract sensor/actuator interfaces.
2) Real-time simulation, in which simulators are replaced by a real-time execution engines while leaving the models unchanged.
3) Hardware-in-the-loop (HIL) simulation in which the environment model is simulated by a DEVS real-time simulator on one computer while the model under test is executed by a DEVS real-time execution engine on the real hardware.
4) Real execution, in which DEVS models interact with the real environment through the earlier established sensor/actuator interfaces that have been appropriately instantiated under DEVS real-time execution.

Model continuity reduces the occurrence of design discrepancies along the development process, thus increasing the confidence that the final system realizes the specification as desired. Furthermore, it makes the design process easier to manage since continuity between models of different design stages is retained.

3.4 Department of Defense Architectural Framework (DoDAF)

The Department of Defense (DoD) Architectural Framework (DoDAF), Version 1.0 (2003), defines a common approach for DoD architecture description development, presentation and integration. The framework enables architecture descriptions to be compared and related across organizational boundaries, including Joint and multinational boundaries.

DoDAF is an architecture description and it does not define a process to obtain or build the description. The Deskbook provides one method for development of IT architectures that meet DoDAF requirements, focusing on gathering information and building models required to conduct design and evaluation of an architecture. The DoDAF defines three elements for any architecture description. These are:

1. Operational View (OV)
2. System Views (SV)
3. Technical Views (TV)

These views provide three different perspectives for looking at an architecture. The emphasis of DoDAF lies in establishing the relationship between these three elements ensuring entity relationships and supporting analysis. The DoDAF approach is essentially data-centric rather than product-centric. The OV, SV and TV are further broken down into specialized views whose brief description can be seen in column 3 in Table 2 ahead.

4 Recent Work and Limitations

According to Zinn [25], the Air Force Chief Architect's office website [38] lists three key impact areas where use of architecture's can provide real benefit:

1. Operations Enhancement
   - 1.1 Requirement Coherence and Prioritization
   - 1.2 Better utilization of fewer personnel
   - 1.3 Deliberate exploitation of innovation
2. Programming and Planning
   - 2.1 IT Investment Decisions (support for POM inputs)
   - 2.2 MIL-Worth Analysis (M&S Executable Architectures)
   - 2.3 AOA Evaluation (Trade Study)
3. Acquisition support
   - 3.1 Enhanced warfighter/user capabilities ID
   - 3.2 Execution Roadmaps
   - 3.3 Source Selection
   - 3.4 Technology application/Transition
   - 3.5 Test Support (MOE/MOP)
   - 3.6 Interoperability and Integration assurance

Dr. Alexander Levis, the Chief Scientist of Air Force acknowledges that M&S can provide an integrated solution in evaluation of the designed architectures [28] but there is no explicit guidance on how to achieve it. An executable architecture (referred to by Levis as an executable model) is defined as "use of dynamic simulation software to evaluate architecture models" (DoDAF, 2003:7.3).

4.1 Current Problem Areas

The source for this text is referred largely from Zinn's thesis [25]:

1. There has been little work done in the area of transforming data from an architecture into the simulation model The 3rd Order Analysis as mentioned in DoDAF doesn't outline any specific achievable tasks in this transformation.
2. Current modeling techniques in DoD use the age old differential equation called "Lanchester Equations" technique to calculate causalities and changes in frontlines, which as evidence put it, are not accurate.
3. Colored Petri Nets (CPNs) provide a solution to some extent but they fall short in introducing dynamics in the model running the simulations. The other drawback of using CPNs is that there is no mechanism to specify 'timing' between the states
4. Another problem highlighted by agent based softwares like SEAS is of the absence of any definiteness of interface specifications that could enable data porting from architecture to the model.

DEVS technology proposes solutions to these problem areas in the rest of the paper.

5 Bifurcated Model-Continuity-based Development Process

Combining the systems theory, M&S framework, and model-continuity concepts leads naturally to a formulation of a *Bifurcated Model-Continuity-based Life-cycle Process* for developing and testing military and other software-intensive systems. As illustrated in Figure 2, the process bifurcates into two streams – system development and test suite development – that converge in the system testing phase. The proposed research will seek to support this development process with the DoDAF-to-DEVS mapping and evaluation environment. The Process has the following characteristics:

DoDAF Specifications: As described in greater detail below, DoDAF descriptions in the operational, system, and technical views are created by designers. Although initially ill-formulated, as the process proceeds, iterative development allows refinement of the requirements and

increasingly rigorous formulation resulting from the formalization and subsequent phases.

Formalization by Mapping into DEVS: Concurrently with the formulation or capture of DoDAF specifications, they are formalized as DEVS model components that are coupled together to form an overall Reference Master Model.

Reference Master Model: The master DEVS model serves as a reference model for any implementation of the behavior requirements. This model can be analyzed and simulated with the DEVS simulation protocol to study logical and performance attributes. Using model continuity, it can be executed with the DEVS real-time execution protocol and provides a proof-of-concept prototype for an operational system.

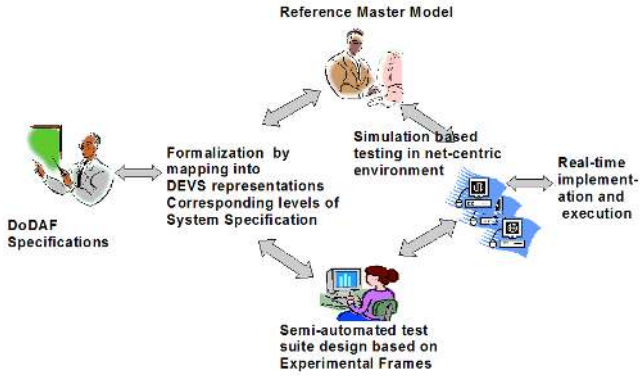

Figure 2: The Bifurcated Model-Continuity-based life-cycle process

Semi-automated test suite design: Branching in the lower path from the formalized specification, we can develop a test suite consisting of experimental frames called test models that can interact with a System Under Test (SUT) to test its behavior relative to the specified requirements. .

Simulation based testing: The test suite is implemented in a Net-centric simulation infrastructure and executed against the SUT. The test suite provides explicit pass/fail/unresolved results with leads as to components/ that might be sources of failure.

Optimization and Fielded execution: The reference model provides a basis for correct implementation of the requirements in a wide variety of technologies. The test suite provides a basis for testing such implementations in a suitable test infrastructure. Test tools should carry into the fielding and operational tests of the system, and provide operationally realistic test cases and scenarios.

Iterative nature of development: The process is iterative allowing return to modify the master DEVS-model and its DoDAF precursor requirements specification. Model continuity minimizes the artifacts that have to be modified as the process proceeds.

6 DoDAF-to-DEVS and the Bifurcated Model-Continuity-based Life-cycle Process

The design methodology provides a process (Figure 3) to transform the DoDAF description of an architecture DEVS representation supporting evaluation and recommendations for a feasible design. Briefly described steps are as follows:

1. The architecture specifications are presented in DoDAF description format as Operational Views, System Views and Technical Views using a supported design language such as CPN or UML
2. The system specifications are then mapped to DEVS specifications according to the translation described in Table 1 that maps the DoDAF views to corresponding DEVS elements. The mapping is illustrated with UML elements and is expressed in XML [20].
3. Test suites for implementations of the design are developed in the test develop stream.
4. Simulation results and their analysis provide the recommendations for a feasible design.
5. Components are developed from the models using Model-continuity principles and the design is verified by the Technical View specifications developed earlier as a part of DoDAF process.

Creation of DEVS Model Repository and DEVS Test Suite occurs in concurrent manner. DEVS Repository serves as a collection of models that are used to develop scenarios, experimental frames and conduct other simulation oriented analysis. DEVS Test Suite is designed to ensure that the required behavior as expressed in input-output pairs is correctly implemented when integrated in the system with timing constraints.

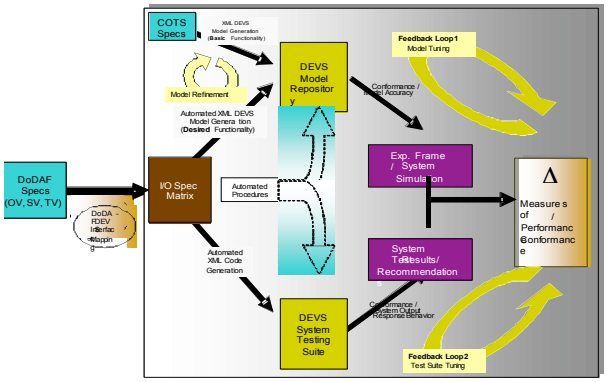

Figure 3: Bifurcated DEVS-to-DODAF Development Process

Analysis of the Experimental frame simulations and the System Test results are compared and evaluated to determine departure from required behavior. This error margin is called the ***Conformance Measure***. Ideally the designed model has a 100% conformance with the Test Suite. If the departure exceeds a given tolerance, the model is revised to increase the model-test conformance. All this assumes that the initial DoDAF specifications have been cast in stone. Typically however, the iterative process will also suggest new or modified specifications at the DoDAF level. The iterative loops can be seen in Figure 3.

Finally, when the models conform to the system test specifications, the Test Suite presents the design and performance recommendations as the outcome of this data-centric process. The Model Repository serves as the basis of design of components based on Model-continuity principles and the Test Suite serves as the benchmark for performance evaluation and matching the Technical specifications as developed in the Technical View DoDAF description

DoDAF Elements			**UMLElements**	**DEVS Elements (generated using XML)**	
	Name	**Description**			
Operational View	OV-1	Top-level Operational View	• Use-case Diagrams	• Top level entity structure	DEVS Model Repository
	OV-5	Operational Activity Model	• Use-case • Activity-Sequencing Diagrams • Data-Flow Diagrams	• Input-output pairs • Port Identification	
	OV-6	Operational Timing and Sequencing Diagrams	• Timing-Sequencing Diagrams • State-machine Diagrams	• DEVS Atomic Model Creation (Initialize Function, internal and external, transition functions, time advance and output functions) for Activity Components • Entity identification	
	OV-2	Operational Node Connectivity	• Logical Components	• Coupling Information • Hierarchical component organization	
	OV-3	Operational Information Matrix		• Input-Output Transaction Pairs • Message Formats	DEVS System-Test Suite
	OV-7	Logical Data Model	• Packages (only for xUML)	• Entity identification • Hierarchical Structure	
	OV-4	Organizational Relationship Chart		• Entity identification • Hierarchical entity structure	DEVS Model Repository
System View	SV-4	System Functional Description	• Use-case Description		
	SV-5	System Functional Traceability Matrix (Based on OV-5)		• Coupling Info Refinement	
	SV-10	System State Description and Event Trace (based on OV-6)		• DEVS atomic model transition functions refinement	
	SV-6	System Data-Exchange Matrix		• Input-Output pair refinement	
	SV-1	System Interface Description (based on OV-2)		• Port assignment Refinement • Entity refinement	
	SV-2	System Communication Description	• Deployment Diagrams	• Coupling Info Refinement (hierarchical management)	

	SV-7	System Performance Parameters Matrix			DEVS System-Test Suite
	SV-3	System-Systems Matrix		• Hierarchical model organization • Entity refinement	DEVS Model Repository
Technical View	TV-1	Current Standards	• Timing Response	• Basic DEVS model for COTS components	
	TV-2	Future Standards		• Improved DEVS model for desired Functionality	

Table 2: DoDAF-DEVS Translation Table

7 Discussion

Currently there is a dearth of existing DoDAF examples to serve as benchmarks for evaluation of the proposed extensions. Since this framework is very much new and still in the development phase, getting a realistic DoDAF version of any architecture at this juncture proved to be a futile exercise. Zinn's work compiles information (essentially a manual task) form OV-6a and OV-5 into a text file that could then used for XML parsing for model creation or feeding into SEAS. We take off from where Zinn [25] left off by giving more structure to the 'compilation' process and how architecture UML specification can be used directly to create DEVS models.

The following table puts the above discussion into more perspective. Desired M&S capabilities are taken from the AFCAO list mentioned in Section 4. Even though it has been realized that M&S is necessary in performing evaluation and developing acquisition strategy, there is more opportunity for current simulation technology to help.

AFACO Reference	Desired M&S Capability	Current working tools (Agent-based or CPNs)	Solutions provided by DEVS technology
1.1	**Requirement Coherence and Prioritization**	No formal methodology exists in defining architectures wherein the data-model can be put directly to use in simulation modeling	The present work aims to accomplish this, by injecting requirements quite early in the design stage of DoDAF architectures, specifically in OV phase.
2.2	**MIL-Worth Analysis (M&S Executable Architectures)**	Work is ongoing in this area. Due to the limitations of the technologies being used, the desired 'execution' is not possible	DEVS provide the capability to: 1. Control simulation on-the-fly [21] 2. Reconfigure simulation on-the-fly [26] 3. Provide dynamic variable-structure component modeling [22][26] 4. Separate model from the act of simulation itself which can be executed on single or multiple platforms using DEVS/HLA [6] 5. Simulation Architecture is layered to accomplish the technology migration or run different technological scenarios [10][23] 6. With its Bi-furcated process automated test generation is integral to this methodology [27]
3.1	**Enhanced Warfighter/user capabilities**	1. Deterministic CPNs 2. Stochastic SEAS but too rigid to reconfigure on the fly 3. Agent-based methodology again falls short in variable structure simulation model	
3.2	**Execution Roadmaps**	No capabilities to control the ongoing simulation to steer it in the 'right' direction	
3.3	**Source Selection**		
3.4	**Technology Application /Transition**	No dynamic reconfiguration of model and simulation reported. The simulation architecture itself has to be layered enough to accomplish technology transition	
3.5	**Test Support**	Agent Based technology, essentially Zinn's work is in this direction CPNs not capable of	

		Automated test generation	
3.6	**Interoperability and Integration Assurance**	Limitations of the methodology itself. No mechanisms reported so far	

Table 3: Comparison of current technologies in development with DEVS on addressing M&S issues in AFACO

8 Conclusions

Under a DoD mandate, DoDAF specifications will become the basis for all information system design in the near future. Although the current DoDAF specification provides an extensive methodology for system architectural development, it is deficient in several related dimensions – absence of integrated modeling and simulation support, especially for model-continuity throughout the development process, and lack of associated testing support. To overcome these deficiencies, we described an approach to support specification of DoDAF architectures within a development environment based on DEVS-based modeling and simulation. The result is an enhanced system lifecycle development process that includes model-continuity based development and testing in an integral manner.

The present work is to be extended towards development of a complete methodology to transform any DoDAF-UML specification to its corresponding DEVS model. The future work will consist of detailed analysis of DoDAF Operational Views and integrated M&S based on idea presented in this paper.

References

[1] DoD Architecture Framework, Software Productivity Consortium, http://www.software.org/pub/architecture/dodaf.asp, last accessed Jan 9, 2005.

[2] DOD Instruction 5000.2 "Operation of the Defense Acquisition System," 12 May 2003.

[3] Chairman, JCS Instruction 3170.01D "Joint Capabilities Integration and Development System," 12 March 2004.

[4] Chairman, JCS Instruction 6212.01C "Interoperability and Supportability of Information Technology and National Security Systems," 20 November 2003

[5] DoD Metadata Registry and Clearinghouse, http://diides.ncr.disa.mil/mdregHomePage/mdregHome.portal/, last accessed Jan 9, 2005

[6] B. P Zeigler, H. Praehofer, T. G. Kim, "Theory of Modeling and Simulation", Academic Press, 2000

[7] Discrete Event Modeling and Simulation Technologies: A Tapestry of Systems and AI-Based Theories and Methodologies Editors: Hessam S. Sarjoughian , François E. Cellier, Spring-Verlag, NY, **2001.**

[8] B. P.Zeigler, DEVS Today: Recent Advances in Discrete Event-based Information Technology, MASCOTS Conference, 2003

[9] http://www.acims.arizona.edu/SOFTWARE/software.shtml, last accessed Jan 12, 2005

[10] H. Sarjoughian, B. Zeigler, and S. Hall, "A Layered Modeling and Simulation Architecture for Agent-Based System Development", Proceedings of the IEEE 89 (2); 201-213, 2001

[11] Cho, Y., B.P. Zeigler, H.S. Sarjoughian, Design and Implementation of Distributed Real-Time DEVS/CORBA, IEEE Sys. Man. Cyber. Conf., Tucson, Oct. 2001.

[12] KH Kim and WS Kang, "A Web Services-based Distributed Simulation Architecture for Hierarchical DEVS Models", AIS conference Oct. 2004, Jeju, Korea

[13] K. Kim, Y. Seong, T. Kim, and K. Park, "Distributed Simulation of Hierarchical
DEVS Models: Hierarchical Scheduling Locally and Time Warp Globally," Transactions of the Society for Computer Simulation International, vol. 13. no. 3, pp. 135{154, 1996.

[14] K. Kim and W. Kang, "CORBA-based, Multi-threaded Distributed Simulation of Hierarchical DEVS Models: Transforming Model Structure into a Non-Hierarchical One," LNCS vol. 3046, pp. 167{176, 2004

[15] Davis K. P. and Anderson A. R. (2003). *Improving the Composability of Department of Defense Models and Simulations*, RAND Technical report (Appendix D).

[16] X. Hu, and B.P. Zeigler, " Model Continuity in the Design of Dynamic Distributed Real-Time Systems", accepted by *IEEE Transactions On Systems, Man And Cybernetics— Part A: Systems And Humans*

[17] Model Driven Architecture (MDA), OMG Document number ormsc/2001-07-01, 2001

[18] A. Newman, S. M. Shatz, and X. Xie, "An Approach to Object System Modeling by State-Based Object Petri Nets," *Int. Journal of Circuits, Systems, and Computers*, Feb. 1998, Vol. 8, No. 1, pp. 1-20

[19] Bowman, Howard, Derrick, John, *Formal methods for distributed processing: a survey of object-oriented approaches*, Cambridge University Press, 2001

[20] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI," IEEE Internet Computing, vol. 6, no. 2, pp. 86{93, March-April 2002.

[21] S. Mittal, B. P. Zeigler, "Dynamic Simulation Control with Queue Visualization", Summer Computer Simulation Conference SCSC'05, Philadelphia, July 2005

[22] X. Hu, B. P. Zeigler, S. Mittal, "Dynamic Configuration in DEVS Component-based Modeling and Simulation", SIMULATION: Transactions of the Society of Modeling and Simulation International, November 2003

[23] S. Mittal, B. P. Zeigler, "Modeling/Simulation Architectures for Autonomous Computing", Autonomic Computing Workshop: The Next Era of Computing, January 2003

[24] B. P. Zeigler, S. Mittal, "Modeling and Simulation of Ultra-large Networks: Methodology Responds to Challenges", ULN Workshop, November. 2001

[25] A. W. Zinn, "The Use of Integrated Architectures to support Agent Based Simulation: An Initial Investigation", MS Thesis, AFIT/GSE/ENY/04-M01, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 2004

[26] S. Mittal, "Dynamic Simulation Reconfiguration and Control in DEVS-Based Component Models", Journal of Defense Modeling and Simulation, submitted 2005.

[27] B. P. Zeigler, D. Fulton, P. Hammonds, J. Nutoro, "Framework for M&S-Based System Development and Testing in Net-centric Environment,", to appear in ITEA Journal, November 2005

[28] A. Levis, L. Wagenhals, " C4ISR Architectures I. Developing a Process for C4ISR Architecture Design", System Architectures Lab, C3I Center, MSN 4D2, Geroge Mason University, July 2000

[29] https://cao.hanscom.af.mil/af-cio.htm