DEVS-Based Dynamic Model Reconfiguration and Simulation Control in the Enhanced DoDAF Design Process

Saurabh Mittal

Eddie Mak

Arizona Center of Integrative Modeling
and Simulation (ACIMS)
ECE Department, University of Arizona
Tucson, AZ 85721
[saurabh, emak]@ece.arizona.edu

James J. Nutaro
Oak Ridge National Laboratories
Oak Ridge, TN 38731
nutarojj@ornl.gov

The combination of DoDAF operational views, which capture the requirements of an architecture, and System views, which provide its technical attributes, forms the basis for semi-automated construction of simulation models. In this paper, we describe an enhanced Model-View-Controller paradigm that works in tandem with the DEVS M&S framework. We also employ the recently introduced DoDAF extensions that incorporate new operational views to allow DoDAF specifications to be written in the Discrete Event System Specification (DEVS) formalism and, in the process, refine these new extensions. This paper describes a DEVS-based network modeling and simulation environment with dynamic reconfiguration and simulation control. The DEVS modeling and simulation framework with its separation of model, experimental frame, and simulator facilitates the development of a simulation framework supporting run-time simulation tuning. We present a layered simulation architecture that provides the capability to control and reconfigure simulation on-the-fly and steer it toward the desired performance parameters. The rapid feedback cycle supported by "real-time" intervention allows experimentation with parameters and structures and results in effective model configuration that is difficult to achieve when turnaround requires hours or days. We explore the area of system reconfiguration further by providing enhanced capabilities to control the parameters for system design and performance evaluation with respect to Net-Ready Key Performance Parameters (NR-KPP) and the way in which the refined DoDAF extensions can expedite the development of Key Interface Profile (KIP) for any DoDAF architecture. We demonstrate the enhanced capabilities with an example of the development of a simulation environment for the Systems Capable of Planned Expansion (SCOPE) command.

Keywords: DEVS, MVC, NR-KPP, KIP, DoDAF, operational view (OV), system view (SV), MoE, SCOPE command

1. Introduction

In an editorial [1], Carstairs asserts an acute need for a new testing paradigm that could provide answers to several challenges described in a three-tier structure. The lowest level, containing the individual systems or programs, does not present a problem. The second tier, consisting of systems of systems in which interoperability is critical, has not been addressed in a systematic manner. The third tier, the enterprise level, where joint and coalition operations are conducted, is even more problematic. Although current test and evaluation (T&E) systems are approaching adequacy for tier-two challenges, they are not sufficiently well integrated with defined architectures focusing on interoperability to meet those of tier three. To address mission thread testing at the second and third tiers, Carstairs advocates a collaborative distributed environment (CDE), which is a federation of new

JDMS, **Volume 3, Issue 4, October 2006 Pages 239–267**

and existing facilities from commercial, military, and not-for-profit organizations. In such an environment, modeling and simulation (M&S) technologies can be exploited to support model-continuity [2] and model-driven design development [3], making test and evaluation an integral part of the design and operations life-cycle.

The development of such a distributed testing environment would have to comply with recent Department of Defense (DoD) mandates requiring that the DoD Architectural Framework (DoDAF) be adopted to express high-level system and operational requirements and architectures [4–7]. Unfortunately, DoDAF and DoD net-centric [8] mandates pose significant challenges to testing and evaluation 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.

This paper begins by providing an overview of the current DEVS technology and the way in which DEVS is positioned to address the need for a new DoDAF-based and net-centric paradigm for test and evaluation at the system-of-systems and enterprise systems levels. Our earlier work [9] enhanced the DoDAF by proposing a methodology to map DoDAF descriptions to DEVS specifications, i.e., DoDAF-to-DEVS mapping. Since DEVS environments such as DEVS JAVA, DEVS-C++, and others [10] are embedded in object-oriented implementations, they support 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) [11]. Moreover, DEVS environments, such as DEVS JAVA, 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 the DoD's HLA standard, CORBA, SOAP, and others and can be readily interfaced to other engineering and simulation and modeling tools [10, 12–17]. Furthermore, DEVS operation over web middleware (SOAP) enables it to fully participate in the net-centric environment of the Global Information Grid [8]. As a result of recent advances, DEVS can support model continuity through a simulation-based development and testing life cycle [2]. This means that the mapping of high-level DoDAF specifications into lower-level DEVS formalizations enables such specifications to be thoroughly tested in virtual simulation environments before being easily and consistently transitioned to operate in a real environment for further testing and fielding.

In [18], Mittal proposed extensions to the DoDAF by introducing two new Operational View documents, OV-8 and OV-9, which allow modeling and simulation to be a critical part in the design process. He demonstrated how DoDAF-DEVS mapping can actually take place from the existing DoDAF UML specifications and how standardized model repositories can be created. The present work aims to refine these new OV-8 and OV-9 documents by providing more details throughout the development process and discussing the use of these documents with *Net-Ready Key Performance Parameters* (NR-KPP) and DoDAF System Views. NR-KPP is instrumental in setting the baseline performance for DoDAF architectures [19]. In particular, we will focus on testing the interoperability of existing architectures and the way in which different architectures can be benchmarked using M&S. We propose an enhanced version of the Model/View/Controller (MVC) pattern [20] that provides dynamic simulation reconfiguration and simulation control using variable structure DEVS-based component modeling [21, 23]. We illustrate the benefits of the proposed enhanced MVC through a current project that deals with development of a simulator for the Systems Capable of Planned Expansion (SCOPE) command.

1.1 Brief Overview of Capabilities Provided by DEVS

To provide a brief overview of the current capabilities provided by DEVS, Table 1 outlines how it could provide solutions to the challenges in net-centric design and evaluation.

We will show how the DEVS technology provides an encompassing framework for test and evaluation (T&E). With its underlying systems theoretical approach, DEVS has the necessary depth and breadth to become the preferred T&E platform for net-centric analysis and design. Section 2 discusses the key component technologies inherent in DEVS, principles of model continuity, and the DoDAF itself. Section 3 mentions the DoDAF specifications, both the original and the extended versions that introduced OV-8, OV-9, and model repository. Section 4 describes the problems and limitations of existing simulation frameworks and their failure to provide any real-time simulation control and component variation. Section 5 deals with the MVC paradigm and its enhanced version. Section 6 presents the layered simulation architecture that is at the core of the introduced technology and how it can be viewed in a distributed

Table 1. DEVS on addressing M&S issues

Desired M&S Capability	Solutions Provided by DEVS Technology
Requirement coherence and prioritization	1) Control a simulation on the fly [21]. 2) Reconfigure a simulation on the fly [22]. 3) Provide dynamic variable-structure component modeling [22, 23]. 4) Separate a model from the act of simulation itself, which can be executed on single or multiple distributed platforms [11]. 5) Simulation architecture is layered to accomplish the technology migration or run different technological scenarios [16, 24]. 6) With its bifurcated test and development process, automated test generation is integral to this methodology [25]. 7) Provide dynamic simulation tuning, interoperability testing and benchmarking [22]. 8) Provide rapid means of deployment using model-continuity principles and concepts like "simulation becomes the reality" [2].
MIL-worth analysis (M&S executable architectures)	
Enhanced user capabilities	
Execution road maps	
Source selection	
Technology application/transition	
Test support including vulnerability analysis	
Interoperability and integration assurance	
Hierarchical modular construction of models aiding system-of-systems testing	
Provide collaborative distributed environment for M&S	

HLA communication network. It addresses the basic requirement for a T&E framework—that the communication network be "always on." It presents ideas on how an M&S framework can provide a means to analyze online networks. It also presents various other constructs that aid the user to control the simulation. Section 7 discusses the concept of dynamic model reconfiguration using DEVS variable structure methodology. It introduces the concept of steady-state simulation that is guided by the end goal in mind. Section 8 discusses the underlying DEVS simulation protocol and how dynamic control is accomplished. It also introduces the concept of hierarchical parameters that find their way in OV-8 and OV-9 documents. Section 9 discusses an example of a complete distributed HLA modeling framework that models nodes as HLA federates. This framework is then used to benchmark different commercial RTIs and to explain how models are "tuned." Finally, Section 10 presents the conclusions.

2. DEVS System Specifications

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

2.1 Hierarchy of System Specifications

Systems theory deals with a hierarchy of system specifications that defines levels at which a system may be known or specified. Table 2 shows this hierarchy of system specifications (in simplified form; see [11]).

- 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 input/output (I/O) relation that 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 I/O pairs of level 1 into non-overlapping subsets, with 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 systems, 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

Table 2. Hierarchy of system specifications

Level	Name	What We Specify at This Level
4	Coupled systems	System built up by several component systems that are coupled together
3	I/O system	System with state-space and state transitions to generate the behavior
2	I/O function	Collection of I/O 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 I/O 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

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 of generating the desired behavior (system at level 1).

2.2 Framework for Modeling & Simulation

The *framework for M&S* as described by Zeigler et al. [11] 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 [11] for a detailed discussion.

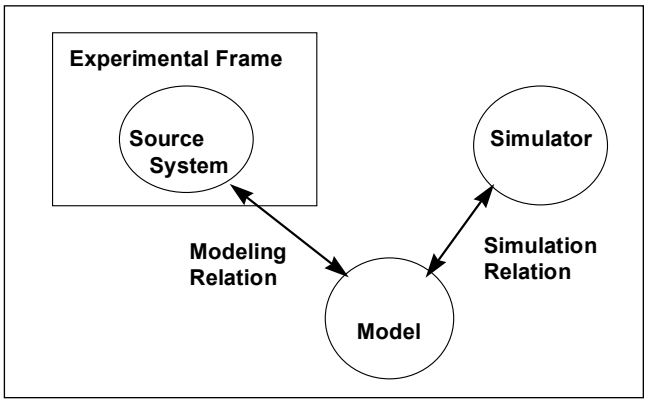

Figure 1. Framework entities and relationships

2.3 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 the opposite of 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 [2]. 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 four steps (see [2]):

1) Conventional simulation to analyze the system being tested 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 engine 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 being tested is executed by a DEVS real-time execution engine on the real hardware;

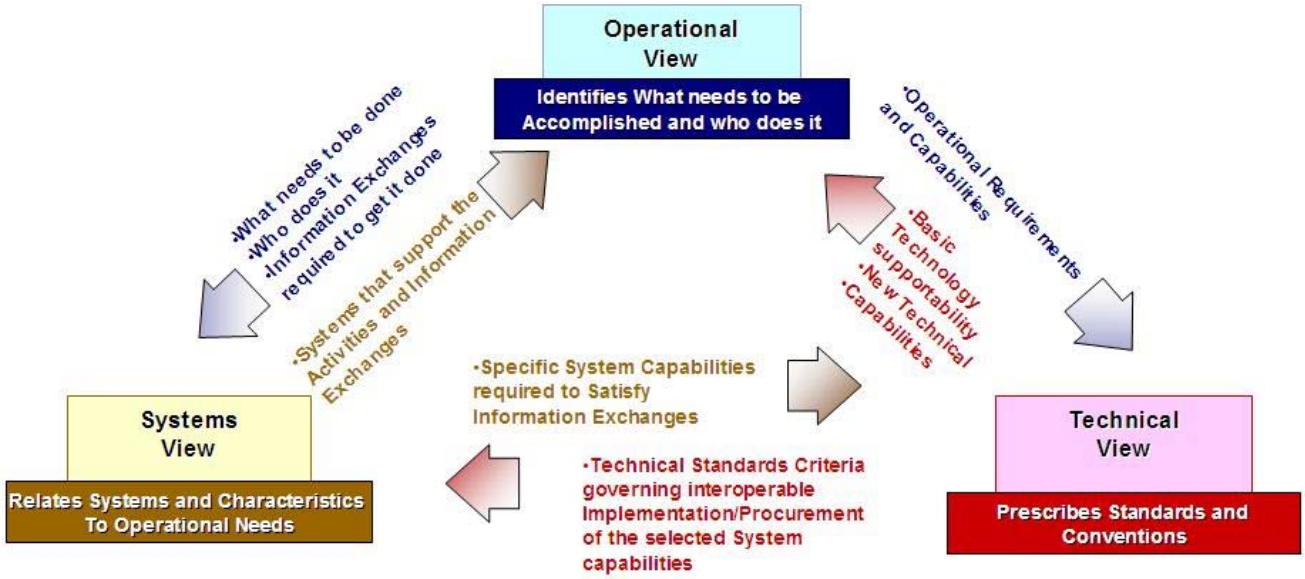

Figure 2. Linkages among views

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. DoDAF Descriptions

3.1 DoDAF Specifications

The Department of Defense Architecture 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 does not define a process to obtain or build the description. The Deskbook [26] 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.

1. Operational Views (OV)

The OV is a description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions. DoD missions include both war-fighting missions and business processes. The OV contains graphical and textual products that comprise an identification of the operational nodes[1] and elements, assigned tasks and activities, and information flows required between nodes. It defines the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges.

2. System Views (SV)

The SV is a set of graphical and textual products that describes systems and interconnections providing for, or supporting, DoD functions. DoD functions include both war-fighting and business functions. The SV associates systems resources to the OV. These systems resources support the operational activities and facilitate the exchange of information among operational nodes. Within this view, how the functionalities specified in OV will be met is elaborated.

3. Technical Views (TV)

The TV is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a conformant system satisfies a specified set of requirements. Within this view, the delivery of

1. Operational node: A node specified in OV that performs one or more operations; a functional entity that communicates with other functional entity to implement a collective functionality or a capability.

systems and functionalities is ensured along with their migration strategies toward future standards.

These views provide three different perspectives for looking at architecture. The interrelationships among these three views can be seen in Figure 2. The emphasis of the 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 3 ahead.

3.2 Extended DoDAF Specifications

Information technology–based systems of the future will be increasingly complex with participants across the globe communicating through disparate channels. Interoperability is very much in question. Scalability and fault-tolerance issues have to be addressed. Capabilities have to be satisfied and reliability has to be ensured. Any large system that DoDAF specification documents intend to build has to realize these important facets of architecture design. Modeling and simulation with its model-continuity principles is fast becoming an accepted method of evaluating design principles ensuring accountability to various components within the system. DoDAF has completely overlooked M&S as a possible means to evaluate design, capabilities, and planned expansion of current architectures. There is no provision for testing the constructed system, either in OV or in SV. The ability to configure systems for optimum performance is not allowed in the current DoDAF specification document.

We have introduced two new operational views, OV-8 and OV-9, that add features to enable M&S of the system under design. More details can be found in [18]. We have also demonstrated how these new documents will be created from the existing operational views. We aim to provide structure to the OV process by shifting the perspective from describing functionality as an activity to an Activity-component with definite interfaces to other Activity components as well as identified entities within an Operational node. To

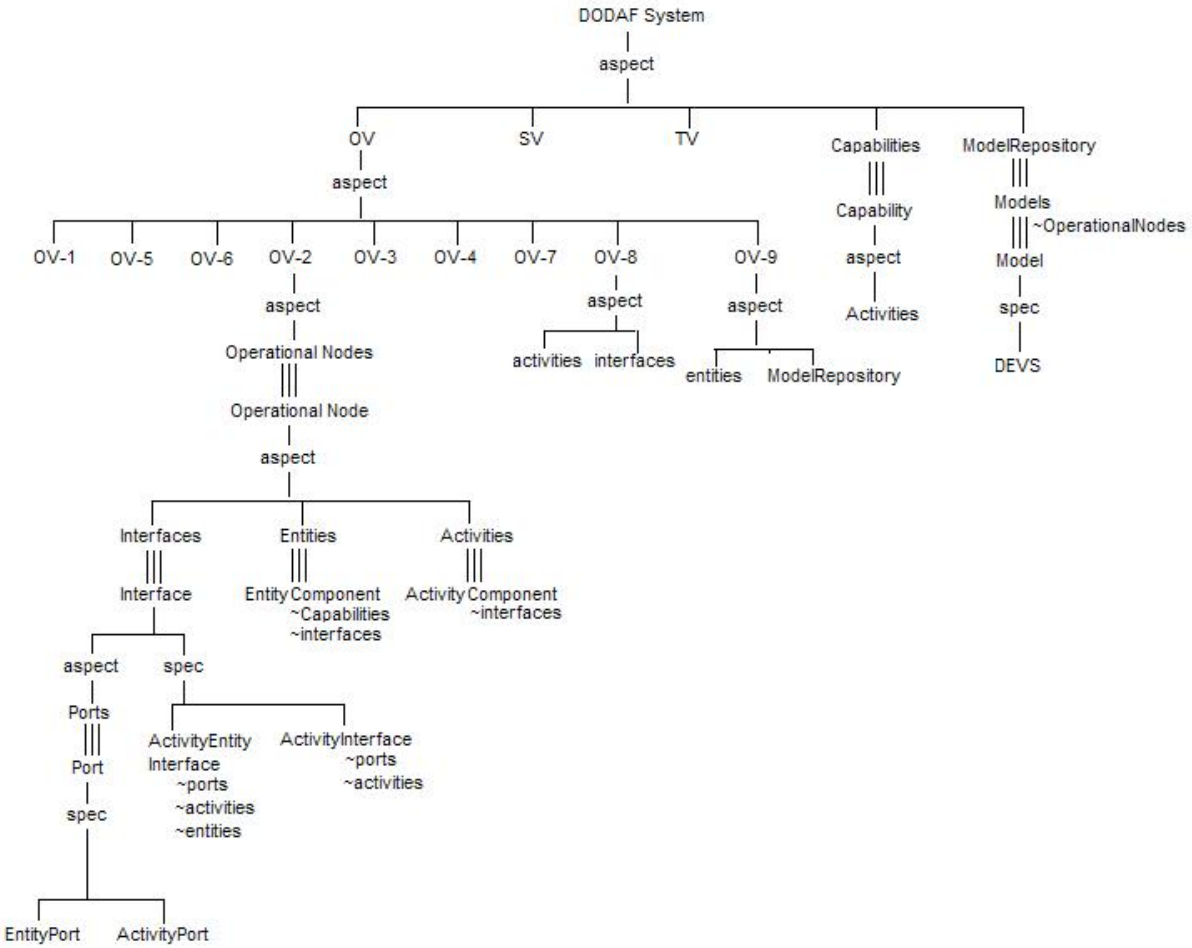

Figure 3. SES for enhanced DoDAF with a focus on OV

what extent an Operational node is decomposable is a subject requiring further research. We have developed a testing process for defined capabilities (that were defined during the conceptual design process in OV-5) and ways in which various rules and doctrines (in OV-6a) can be evaluated for interoperability with different capabilities. By purview of the information contained in OV-9 we have introduced the *model repository* as an important aspect of DoDAF system specification that enhances the DoDAF by making way for M&S activity. Figure 3 shows the *system entity structure* (SES) snapshot of the enhanced DoDAF with focus toward the operational views. (See appendix for more information on SES.)

3.3 Refinement of OV-8

In relation to the DoDAF, as M&S is not mandated in the current handbook; there exists no place to specify the parameters of any kind, other than in Systems View document SV-7. In the current DoDAF, OV deals with functionality; SV deals with system identification, specification and compliance; and TV provides technical feasibility considerations. However, taken altogether, there is no place where the designer can refer to a "significant parameter set." It is not considered in the OV process and not discussed at all until systems that perform the functionality are identified in SV-4 and SV-5. The architecture designed as a result of the current process is deemed to be too rigid to allow for design exploration (through M&S), let alone any boundary conditions for technology transition studies.

In our earlier work [18] we had proposed two new operational views, OV-8 and OV-9, dedicated to M&S. OV-8 deals with the identification of Activity-components and OV-9 deals with association of these Activity-components to specific system and entity components within an operational node. We refine these two documents here by adding one more column in both (see Table 3). The additional column relates to the "significant" parameter that is worthy of the system designer's attention during construction of Systems View. These significant parameters will, in fact, be admitted in to the experimental frame to control the simulation study.

3.4 NR-KPP and OV-8

Net-Ready Key Performance Parameters (NR-KPP) are a key to measuring the readiness for transformation into a fully interoperable and secure net-centric warfare environment. As currently stated in the Joint Staff Guidance,

> The NR-KPP assesses information needs, information timeliness, information assurance (IA), and the net-ready attributes required for both the technical exchange of information and the end-to-end operational effectiveness of that exchange. It consists of verifiable performance measures and the associated metrics required evaluating the timely, accurate and complete exchange and use of information to satisfy information needs of a given capability. It is composed of the following four pillars:
>
> - Compliance with the Net-Centric Operations and Warfare Reference model (NCOW-RM)
> - Compliance with applicable Global Information Grid (GIG) Key Interface Profiles (KIP)
> - Verification of compliance with DoD IA requirements; and
> - Supporting integrated architecture products required to assess information exchange and use for a given capability.

Integrated architectures are the most critical components of the NR-KPP because they establish both the operational and systems context for information exchange. They are developed and documented using the DoDAF, which provides templates for the 27 distinct views. Two other critical components in NR-KPP, namely, the NCOW-RM and KIPs are in the evolving state. KIP tools help standardize and manage interfaces to networks,

Table 3. Enhanced OV-8 allowing specification of significant parameters for an activity

S. No.	Activity ID Component	Significant Parameters	Connection ID	Source Activity	Input Interface Name	Message Des.	Op. Node	Source Document/ Diagram
1	A6						O1	
2	A6.1		CA6.1	A6.11	inSigMovY	AMT	O1	Figure 12/ OV-6b, c
3	A6.2		CA6.2	A6.11	inSig-MovN	Target	O1	Figure 12/ OV-6b, c
4	A6.3		CA6.3	A6.1	inTrkData	Data	O1	Figure 12/ OV-6b, c

services, and communication pathways. Initial attempts to manage interfaces required identification of specific interfaces between each and every system for which an operational interchange of information was required. The net-centric vision requires a paradigm shift from "one-to-one" relationships to "one-to-many" relationships, and KIPs are the key profiles enabling that shift. The KIP consists of refined OVs and SVs, and interface control document/specification (ICD), an engineering management plan, a configuration management plan, a TV with an SV-TV bridge, and procedures for standards conformance and interoperability testing [27]. The Joint Staff categorizes the key interfaces in four broad categories, viz., communications computing, enterprise services, and network operations (NetOps), and other explicit interfaces (in addition to seventeen mentioned since first draft). The final version is referred as in [7], released March 2006. From an evaluation perspective as has been recognized, if the KIP is not defined then it clearly cannot be tested. As a part of overall T&E strategy, the NR-KPP T&E strategy [27] is twofold: first, identify and gather sufficient data from developmental events to verify that the key subcomponents for the NR-KPP have been satisfied, and verify that the information and systems data exchanges can be accomplished; and second, validate these exchanges to achieve the mission in an operational test (OT) environment.

The two steps mentioned in the overall strategy relate to the SV-6 document that specifies the system data-exchange matrix. The first step verifies the data exchange, while the second step validates it in the operational test environment; see Figure 4. The OV-8 and OV-9 documents lie in the operational test domain. As stated earlier, the current DoDAF OVs do not provide any mechanism to facilitate M&S; these new OVs, in addition to specifying interfaces, are refined to identify these operational parameters for any particular Activity-component. As the KIP specifies an *interface control document* (ICD), OV-8 expedites its development by providing ready operational interfaces, in relation to a particular capability, that can be mapped to specific system interfaces with relation to SV-6 and SV-7. Providing the mapping for SV-6 and SV-7 is outside the scope of this paper. The objectives of proposing OV-8 and OV-9 have been to empower the DoDAF with M&S

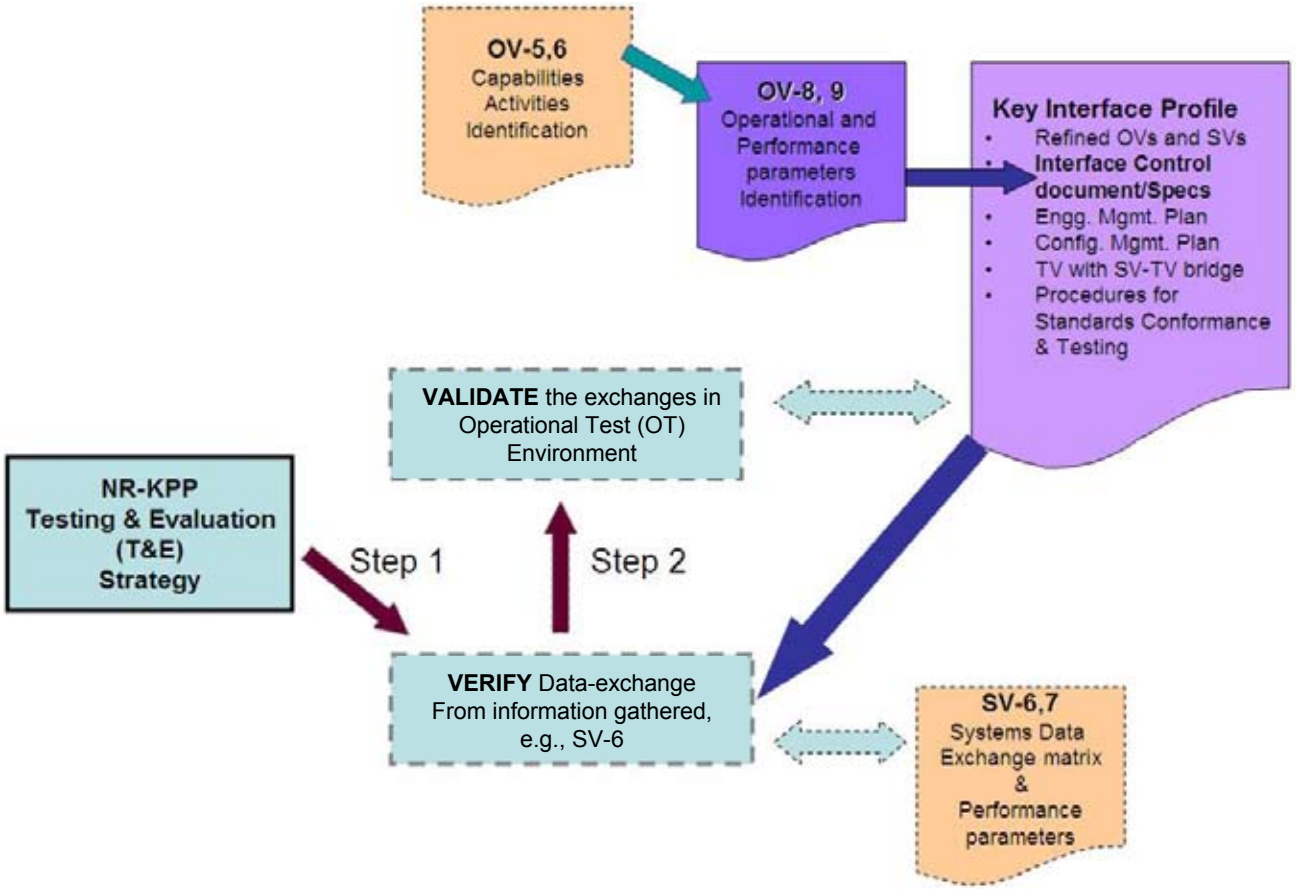

Figure 4. Role of OV-8 in construction of Key Interface Profile (KIP)

at the operational development stage and to show how different capabilities can be composed from various independent activities. Proving interface specification and allowing composition using this interface specification in OV-8 is a key element of this proposition. The refined OV-8 allows for specification of parameters, for both an Activity as well as the composed capability that will be mapped to the NR-KPP with reference to any particular DoDAF architecture.

4. Problems Associated with Current Simulation Frameworks

Although a number of commercial and academic simulators are available for complex network studies, none have the capability to tune the simulation while it is in execution. Due to tight coupling between the network model and the simulation engine in such simulators, the capability to introduce changes in parameter values during execution is limited or non-existent. The work described here has the objective of developing a DEVS-based network modeling and simulation environment with dynamic simulation control and queue visualization. The DEVS modeling and simulation framework separates model, experimental frame, and simulator. This modularity facilitates the development of a simulation framework supporting run-time simulation tuning. The motivation behind providing "real-time" intervention is to support a rapid feedback cycle that allows experimentation with network parameters and structures. This can result in an effective network configuration that is difficult to achieve when turnaround requires hours or days. Furthermore, such instantaneous observation and control enables important transient situations to be recognized and considered.

4.1 Real-Time Control and Visualization Limitations of Existing Network Simulators

Some of the limitations of existing network simulation packages are as follows:

- Everything has to be programmed prior to simulating the network.
- User interfaces are not easily customized; they provide largely textual interfaces.
- There is no support for changing parameters and component structures during simulation.
- Simulation run times tend to be long (a few hours); more importantly, if a run ends in a crash, there is no way to intervene and readjust the system.
- There is little run-time visualization of the system behavior to aid understanding and to steer the simulation in a productive direction.
- Model and simulation calibration is a new concept, largely unattended by the legacy and current simulators.
- Model-driven design and development is a new technology supported by only a handful of simulation frameworks.
- Distributed M&S and concepts like model repository are not supported in most of the frameworks.
- Treating an M&S T&E framework as an "online" system by itself is non-existent and unaddressed by current simulators.
- Performance-oriented simulation frameworks are non-existent. Most are bounded by initial model configuration.

To develop a network modeling and simulation environment that addresses these limitations, we extended the existing Discrete Event System Specification (DEVS) software, DEVSJAVA. We discuss the layered architecture underlying the network simulation environment. After describing this architecture, we discuss some proposed run control and visualization techniques intended to greatly improve user understanding of, and ability to control, the complex structural and behavioral relationships characteristic of large network behaviors. An example of the use of the architecture will be given; it concerns modeling and simulation of network-based distributed engineering studies performed using the High Level Architecture (HLA) middleware.

5. Model/View/Controller (MVC) Paradigm and DEVS Framework

5.1 Earlier Work

Nutaro [28] proposed the Model/Simulator/View/Controller (MSVC) paradigm, as an extension of MVC. He promoted the separation of model and simulator and has listed many advantages that come about with this idea, most important being the reuse of simulation software, especially in the context of distributed simulations. The other problems that are solved by this paradigm are as follows:

1) Distributed simulation protocol changes can be encapsulated within the controller (input and time management policies) and viewer (output policies) objects.
2) By separating the viewer and controller it is straightforward to add displays, logging tools,

and other output processing devices to the simulator.

3) Modeling, simulation, distribution or parallelization, and user interface issues can be addressed separately.

Nutaro demonstrated an application of middleware simulation, wherein the simulator was tuned to display the behavior of certain middleware by incorporating effects such as RTI latency (with reference to distributed simulation HLA framework). In his methodology, the simulator is a thread derived from the controller thread that contains the platform (RTI latency) delay parameter. As the controller thread generates this event, it is communicated to the simulator as well as to the viewer using inter-thread communication. Although Nutaro did not consider model updating or model control, his work constitutes a part of our enhanced MVC framework, where there is full capability in the controller to modify the model as well as the simulator.

Our work is implemented in DEVSJAVA and has a super-thread that runs at the root-coordinator level that monitors the experimental frame for any user-generated activity controls. There exists no viewer thread as the viewer objects are created hierarchically as delegated classes of the model as well as the simulator object. Any modification in their state is also reflected in the contained viewer object. The viewer object displays are derived from the *java.awt* package. Consequently, they inherently have independent thread that repaints.

5.2 Enhanced MVC

Figure 5 below provides the graphical representation of an enhanced MVC paradigm. It has been represented with respect to the DEVS M&S framework components. Model and View take their usual functions and meanings. The Control in MVC is explored in more detail and is mapped to the DEVS Experimental frame. Internally, the Experimental frame has a modular structure with a *basic control* component and *controller A* and *B* as derived components.

The *basic control* component translates the information contained in OV documents. It extracts the parameter-set as specified in the refined OV-8, in section 3.3. It also extracts information from the NR-KPP set and integrates the information into a unified controller that the user can refer to. It is specialized into two components, one dedicated to simulator middleware control and the other dedicated to model control. It also assigns different parameters to the

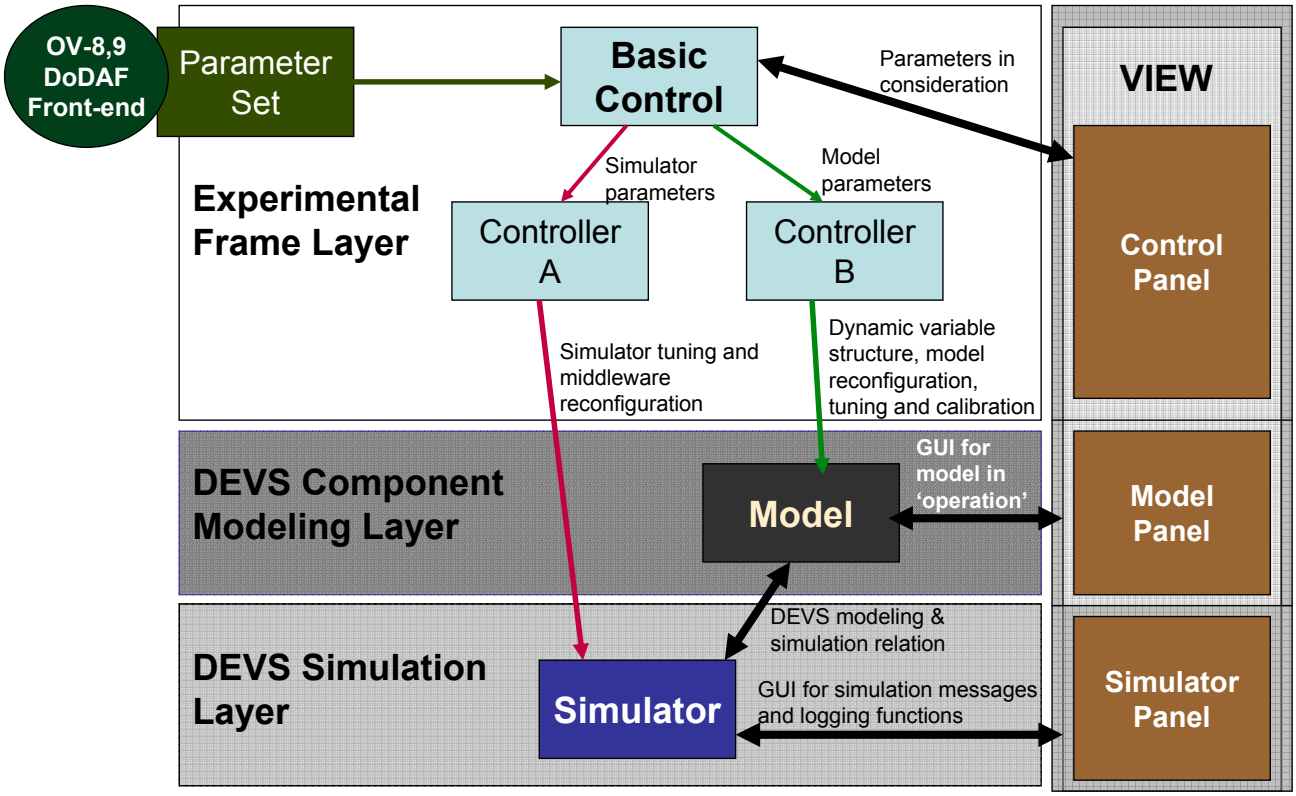

Figure 5. Enhanced MVC paradigm with DEVS M&S framework

appropriate controller. From an earlier work discussed in section 5.1, *controller A* provides tools to control the DEVS simulator, more appropriately the middleware aspect of simulation. *Controller B* provides the toolset to control the model. Details about middleware control can be seen in Nutaro [28]. *Controller B* provides functionality to vary the number of components, in addition to the parameters in a component, both at the component and subsystem level. The parameter set for both the controllers is made available to the user as a sliding bar in the *controller frame* in the View panel that enables the user to tune the active simulation toward optimum performance.

The enhanced MVC has exhaustive control expressed in the experimental frame domain. The Experimental frame component in the DEVS M&S framework is a key construct that enables the user to drive and maneuver the simulation in the "right" direction. The concept of experimental frame, i.e., a mechanism by which an experimental scenario is designed for the model architecture, is further enhanced to enable the user to reconfigure and tune the simulation itself. Benefits of user intervention have already been highlighted in section 5.1 and are explored in more details in section 8. Given that the user has the capability to control the simulation parameters, the issue of extraction and identification of those parameters is taken care by the *basic control* component that interfaces with the DoDAF documents OV-8 and OV-9. Consequently, the Experimental frame now provides rich control equipment that the operational test designer can use to his advantage.

6. Dynamic Simulation Reconfiguration

6.1 Variable Structure DEVS

A component is "a nontrivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. It conforms to and provides the physical realization of a set of interfaces" [29]. A component system is built by composition of various independent components and by establishing relationships among them. As each component has a high degree of autonomy and has well-defined interfaces, variable structure of components can be achieved during run time. For component-based modeling and simulation, variable structure provides several advantages:

1) It provides a natural and effective way to model those complex systems that exhibit structure and behavior changes to adapt to different situations. Examples of these systems include distributed computing systems, reconfigurable computer architectures [30, 31], fault tolerant computers [32], and ecological systems [33]. Structure changing and component upgrading is an essential part of these systems. Without the variable structure capability it is very hard, if not impossible, to model and simulate them, let alone study the transition effect that the system incurs when new components are added in a real deployed system.
2) From the design point of view, variable structure provides the additional flexibility to design and analyze a system under development. For example, it allows one to design and simulate a system in which the components are added or removed incrementally and form dynamic relationships with existing components.
3) It allows one to load only a subset of a system's components during simulation. This is very useful to simulate very large systems with a tremendous number of components, as only the active components need to be loaded dynamically to conduct the simulation. Otherwise, the entire system has to be loaded before the simulation begins.

In general, there are six forms of reconfiguration of component-based systems [34]:

1) Addition of a component,
2) Removal of a component,
3) Addition of a connection between two or more components,
4) Removal of a connection between two or more components,
5) Migration of a component, and
6) Update of a component.

The first two operations result in an update of the modeled system where there is a change in the number of components in the system, the next three result in a reconfiguration of the existing system, and the last one results in the modification of the component itself, either its behavior or its interface structure. In DEVS these are collectively known as variable-structure modeling. More details about said operations can be found in [23].

As variable structure changes a component-based simulation during run time, boundary conditions and the limits to which a component affects other components need to be specified with said operation. With reference to Table 1, the model reconfiguration can be implemented at any of the specified levels. These issues are very well addressed in [23]. The variable structure provides the flexibility to design and analyze a complex hierarchical system under development, as well as during a running

simulation, as supported by the dynamic structure SES capability.

6.2 Implementation of Variable Structure in Extended MVC

Variable structure essentially deals with modification of the component as well as of the number of components that specify the modeled system. Its power lies in its run-time implementation that gives us the capability to study the transition effects when the system is presented with a different number of components and interrelationships. This is entirely a modeling issue and is independent of how the system is simulated when presented with such changes. With the DEVS modeling approach, this is brought to fruition in its modeling layer. With the proposed MVC approach, as is quite obvious, this is implemented in the modeling layer that is in control of the Experimental frame controller layer. The modeling layer that holds the system model, its configuration, and the inter-component relationships receives commands from the Experimental frame on modifying the system. The user is in charge of the Experimental frame. Consequently, if he wishes to modify the system structure he is given the toolbox to modify the model from the experimental frame. Of course, the toolbox is also designed by the modeling designer who decides if the system is to be analyzed and the chosen component plays a significant role in system dynamics and performance. With the closure under coupling property inherent in DEVS formalism, an entire subsystem or an individual component in system can be added as a "component" in the model, in addition to its relationships with other existing components. This property aids in adding a complete system model as a component in a running simulation. With reference to Figure 5, the Experimental frame view will contain the controls that the user can perform to modify the structure of the model.

6.3 Notion of System Steady State

Evolution is a discipline by which one can understand the growth of a "system" with respect to time. Modeling growth is a difficult concept, let aside simulating "growth." Biological evolution is studied through looking into the past and seeing how different species have changed according to their environment. In computer systems, the Internet is one such system that has "evolved" over time and has resulted in a World Wide Web that now sustains heterogeneous components sustaining together. Evidently, no one could foresee during its conception days that it had the potential to become the Internet of today with over one billion hosts. In order to model growth, one has to have the capability to modify the structure of constituent components—its interfaces on how it changes when the component is placed in different environments. Biological organisms survive by a process of adaptation, and transmitting this information to progeny with encoded information unlike the computer systems. The computer systems are characterized by rigid interfaces through which they communicate with the "environment." Certainly we are not focused toward modeling adaptation, though it can be done with the current DEVS suite, but trying to understand the response of system when another component is introduced in the system is of prime importance. The response time of a system is defined as the time taken by the system to display any effect once the model has been modified. There are legacy systems, and the new technology is bringing new components that need to be backward compatible. The situation with respect to IPV4 and IPV6 is one such example in which the communication network has a new standard that needs to be deployed. IPV6 has been around for more than ten years, and according to various sources, it will take another ten years for the current Internet to be completely IPV6 compliant. Testing of IPV6 in conjunction with IPV4 is a big limitation [35]. The analysis of these kinds of situations can be very readily done with the current capability by introducing links and components to the existing network model and observing how the system responds.

The steady state of any network system can be defined as the situation when the computer network is stable and there is constant throughput, network latency, and there are no overflowing buffers in routers. In essence, it boils down to the efficient utilization of bandwidth across all links such that there are no blockages. Total data transmitted from network components is received at the designated destinations, with allowable errors. Consequently, capacity planning is onestudy that results in quantifying the bandwidth in order to make the system stable with a specified number of components. Looking at it in inverse perspective, finding the number of components that can be sustained by any particular deployed network is of equal interest. The question arises: How can we model a network system in which the system can simulate the growth of this network, arriving at a steady state and providing us with the result that the network can sustain a particular number of components? The current variable structure capability provides us with the needed functionality in which the Experimental frame is given the control to "arrive at steady state." What it actually means is: once a small model of the network system is simulated

and utilization is reported, the system continues to keep adding new (preordained) components, along with their relationships, to the existing system until the system reaches a specified network throughput. At what rate the new components are added is a tunable parameter, made available in the experimental frame. This whole exercise shows, given a certain system exhibiting certain behavior, how the system would perform and evolve if let loose, or what the maximum number of components is that the system can be loaded to so that it maintains a steady state! To determine at what result-set the system would break, or if it has a "survivable" nature, is worth conducting analysis. The run-time capability gives us a window to monitor the effects the system incurs when it is modified by external effects like the rate of growth of the system.

7. Dynamic Simulation Control

7.1 DEVS Simulation Engine

DEVS has been erected on a framework that exploits the hierarchical, modular construction underlying the high level of system specifications. The basis specification structure in all the associated DEVS derived formalisms, e.g., DTSS, DESS, is supported by a class of atomic models. An atomic model is an irreducible component in DEVS framework that implements the behavior of a component. It executes the state-machine and interacts with other components using its defined inports and outports. Each such atomic class has its own simulator class. A network of these atomic models constitutes a *coupled* model that maintains the coupling relationships between the constituent atomic components. Each such coupled model class has its own simulator class, known as a *coordinator.* Assignment of coordinators and simulators follows the lines of hierarchical model structure, with simulators handling the atomic-level components and coordinators handling the successive levels until the root of the tree are reached. These simulators and coordinators form the DEVS simulation engine, and they exchange messages by adhering to what is known as DEVS simulation protocol (see Figure 6). The message exchange is depicted in the figure below. For more details about the simulation protocol refer to chapter 8 of [11]. The figure below shows the mapping of a hierarchical model to an abstract simulator associated with it. Atomic simulators are associated with the leaves of the hierarchical model. Coordinators are assigned to the coupled models at the inner nodes of the hierarchy. At the top of hierarchy there is a root-coordinator that is in charge of initiating the simulation cycles (see Figure 7).

Since the DEVS model is based on DEVS formalism that is based on mathematical systems theory, the behavior expressed through DEVS can be translated to any other formalism, though there exist no other theoretical M&S frameworks. With the separation of the model from the simulator, the advantage is that it *supports formalism interoperability*. The next subsection throws light on how an experimental frame intervenes in the DEVS simulation protocol by causing interrupts, and how it implements dynamic simulation control.

7.2 Interrupt Handling

The *controller frame* is built on top of a root coordinator in DEVSJAVA shown in Figure 7. We developed interfaces to enable the DEVS engine to take into account the change of experimental frame parameters during the simulation run. It generates interrupts, which are handled by the coordinator in DEVSJAVA. The event from the controller frame is handled by the root coordinator that holds the simulation at that

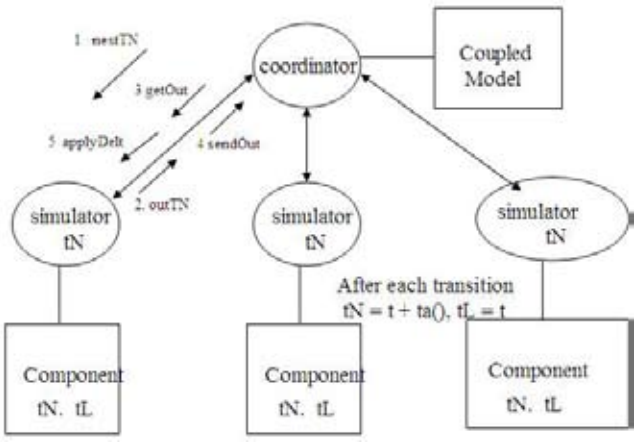

Figure 6. DEVS simulation protocol

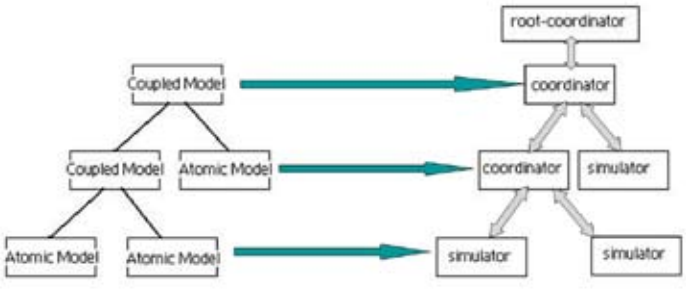

Figure 7. Hierarchical simulator assignment for a hierarchical model

instant, taking care of the simulation state. The event then is channeled through the hierarchical simulator network to the intended model. Once the model has been updated, the root coordinator resumes the simulation by reinitiating the DEVS simulation protocol. Consequently, the model is updated in between the running simulation with other events still being held in different component simulators. Only the intended model is updated, which then participates accordingly as before. How this event (parameter update inside a model) brings change or how the system responds to this change can be seen very well in different visualizers. Examples can be seen in later sections.

7.3 The Notion of "Simulation Control" Explored

Having laid out the framework to implement the dynamic simulation control, we also explored different methodologies in which the simulation can be controlled. Following are the three ways by which the simulation can be interjected and brought to successful execution.

7.3.1 Automated Control

In this methodology, we have stored procedures, basically a predefined event list stored as a file that is being read actively during the running simulation and generates events that sends interrupts to the coordinator. This does not require a *controller frame* that is used to provide real-time interrupts. The *experimental frame* takes the shape of this file in which different scenarios are preloaded along with simulation parameters. Certainly, execution of a scenario can be considered as one simulation run or a session, but the introduction of a parameter set in the experimental frame that is injected dynamically in the running simulation is of prime interest. This approach has been implemented by Nutaro. This methodology is verily extended toward the following setup shown in Figure 8 where the SES family of test cases is implemented as an XML file. The sequence of test is executed in a sequential manner and reported.

7.3.2 Manual Reactive Control

In this methodology, the *experimental frame* is operated through a *controller frame* that is designed by the system test designers. The significant parameters and models are identified with reference to the OV-8 document or NR-KPP set and made available in the controller frame. This methodology provides us with a mechanism to manually interject in the running simulation to introduce modifications. It also provides us with the capability to steer the simulation if the simulation is moving toward a "crash" or if the user wants to see the temporal effects of any parameter update. The capability to steer and study the effects of any single parameter is a powerful capability and is almost nonexistent in current simulators, both in the academic and commercial arenas. There is, however, some software available in the business finance domain that provides this capability. The concept of analyzing parameters in a reactive manner has not been applied to any M&S framework to date. The examples in later sections display this approach.

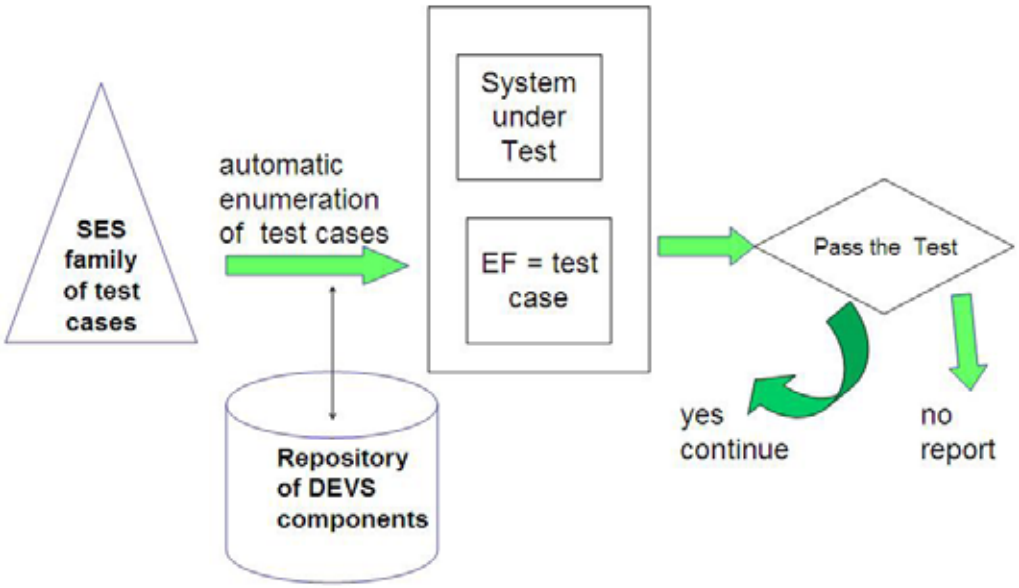

Figure 8. Automated test suite execution

7.3.3 Hybrid Control

As the name suggests, this methodology takes the best of the above two approaches. This methodology has an automated scenario generation/modification capability as well as reactive control through the *controller frame*. The main purpose of the *controller frame* in this approach is to study the temporal effects and steer the simulation toward optimum performance.

7.4 Parameter Control

This subsection presents some ideas on the selection and categorization of parameters. Two classes of parameters that were identified for any system are the tunable parameter set and the result parameter set.

7.4.1 Tunable Parameter Set

This set is comprised of the parameters that are to be included in the Experimental frame. This set is termed "tunable" for obvious reasons, as the simulation analysis is conducted to study their effects on the system performance when their values are modified. These parameters are called tunable parameters because these parameters are implemented as a "slider" component in the controller frame with definite bounds. The user can control this slider to tune the system for optimum performance. In the network system terminology, link capacity, router buffer, etc., can be classified as tunable parameters. With reference to the DoDAF and NR-KPP, this makes more sense, as we need to understand the impact of the identified "significant" parameter on the overall system performance.

7.4.2 Result Parameter Set

This set is comprised of the aggregated result values that provide the overall system performance estimates. SV-7 provides a place where these documents could be found on a per subsystem basis. However, the holistic result parameters still need to find an appropriate place. There should be a dedicated place in the systems view with respect to the overall system performance. The aggregated parameters in a network system can be thought of as latency, network throughput. This parameter takes leverage from the NR-KPP set that is needed to satisfy the baseline system performance. Its mapping with SV-7 is beyond the scope of the current work.

7.5 Synopsis

The above discussion has illustrated how the DEVS simulation framework provides new capabilities in the Experimental frame and how these capabilities are implemented. It also shows that an experimental frame is the place where the user can modify the model and can modulate the simulation according to need. From the basic capability of creating an experimental scenario for the modeled system, we have enhanced it by providing more features like simulation control and parameter tuning. We have also explored various ways simulation control could be performed and how parameters are categorized to find their way in the Experimental frame. Together with the variable structure capability described in section 6, the experimental frame becomes an all-encompassing user interface to a complex hierarchical system model under simulation. It gives the user more power to observe and visualize the simulation by isolation at the parameter level and the component level as well as on the subsystem level.

8. Example to Illustrate the Current DEVS Technology

8.1 Systems Capable of Planned Expansion (SCOPE) Command[2]

SCOPE command is a highly automated, high-frequency (HF) communication system that links U.S. Air Force (USAF) command and control (C2) functions with globally deployed strategic and tactical airborne platforms. SCOPE command replaces existing USAF high-power HF stations with a communication system featuring operational ease of use, dependability, and seamless end-to-end connectivity comparable to commercial telephone services. The network consists of fifteen worldwide HF stations (see Figure 9) interconnected through various military and commercial telecommunications media (see Figure 10). It increases overall operational and mission capabilities while reducing operation and maintenance costs.

The HF radio equipment includes the Collin's Spectrum DSP Receiver/Exciter, Model RT-2200. The radios feature Automatic Link Establishment (ALE) and Link Quality Analysis (LQA) capability and are adaptable to future ECCM waveforms FSK, MIL-STD-188-110B, and STANAG 5066. The transmit subsystem includes 4-kW solid-state power amplifiers, a high-power transmit matrix, and a combination receive/multicoupler antenna matrix. A

2. SCOPE Command description taken from Rockwell Collins website.

Figure 9. Geographic locations of fixed stations

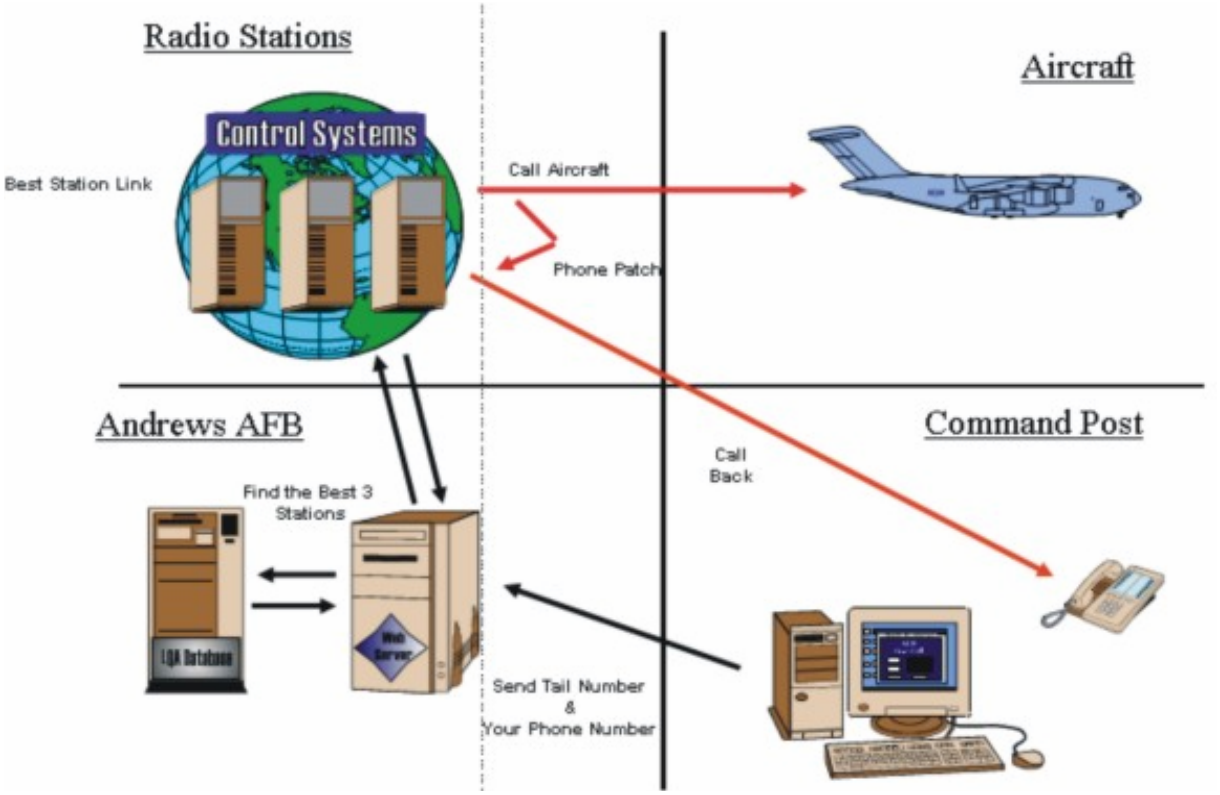

Figure 10. Communication flow diagram for SCOPE command

typical SCOPE command station includes operator consoles (HFNC), circuit switching equipment (DES, DSN, LCO), HF radios (ALEs), RF matrixes (RTs), and antennas (RXs, TXs). A non-blocking digital electronic switch (DES) connects the station to the local military and/or commercial telecommunication services. The switch features unlimited conferencing, modular sizing, a digital switch network, a precedence function, and capacity for up to 2,016 user lines.

SCOPE command uses a modular, open-system design to automatically manage and control all network operations, including those at split-site stations. To achieve maximum flexibility, the system uses commercially available standards-based

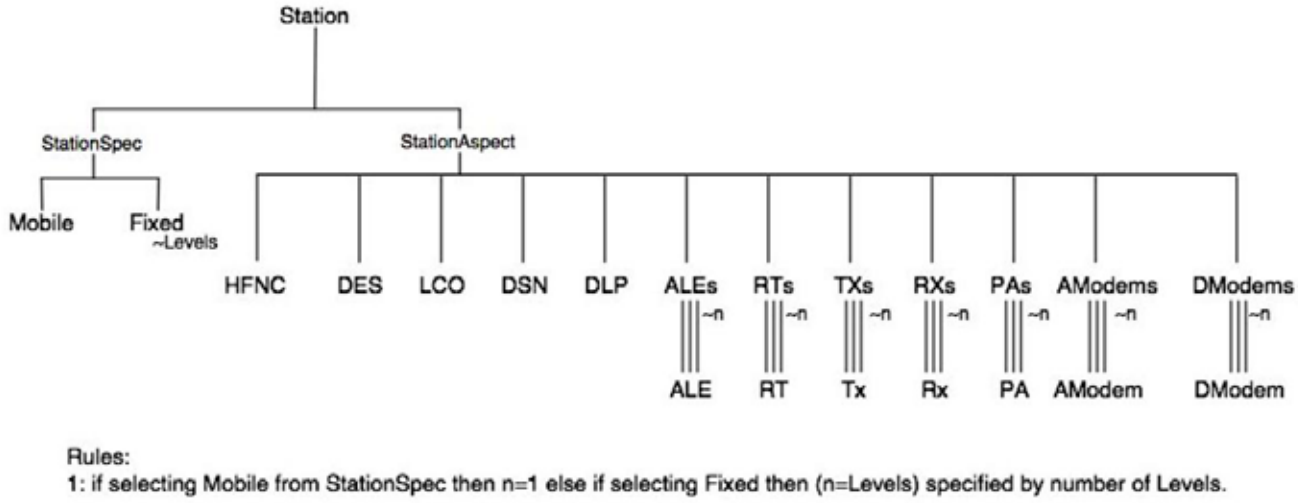

Figure 11. System entity structure for SCOPE command system showing the fixed and mobile (aircraft) stations

software and a multitasking operating system. This approach permits fourteen out of fifteen network stations to operate "lights out" (unmanned) and to be economically controlled from a central location. The control system also includes LAN software, servers, and routers to support unlimited LAN/WAN.

The program includes a Systems Integration Lab (SIL) and test-bed facility located in Rockwell Collins's Texas facility. The SIL is used to predict the impact and risk that any changes or upgrades will have on system performance, integrity, or costs before actual implementation begins. The SIL includes a fully functional SCOPE command station for performing baseline design verification, and interface compatibility and functional verification tests.

Joint Interoperability Test Command (JITC) is the only government agency that is assigned the task to validate and authorize IT systems for military operations [7]. The HF SCOPE command system has also been evaluated by JITC. In collaboration with Dr. Eric Johnson, a simulator was developed in the C language around 1997 that was validated and eventually used by both the government and the industry to conduct experiments and run scenarios. The simulator was an exhaustive and comprehensive effort with respect to the details it implemented and served its purpose well. However, in today's circumstances, the same simulator is obsolete due to the heterogeneous nature of today's network traffic, in which e-mail occupies a considerable percentage of traffic. The simulator is now being upgraded at the ACIMS lab in order to make it more useful for current demands. These demands stem from the possibility of expansion of the current infrastructure of the SCOPE command. Questions arise such as how many stations need be added to service a required workload. Also needing to be investigated are trade-offs such as whether it is more economical to add more stations or increase the number of internal radio levels within stations to meet the anticipated demands. Air traffic has increased manifold since 1997, along with the computing technology. Consequently, the transition effects need to be monitored more closely, and the overall system response time[3] needs to be documented. The significant parameters that have the most impact on system performance have to be identified. To more easily address such questions, an effort is being made to modularize Johnson's 15K lines of code into a component-based structure depicted in Figure 11. Once "componentized," the components are made DEVS compliant resulting in a DEVS-based simulation package to support the systems engineering needs of the SCOPE command.[4]

To study the effect of changes/upgrades introduced to the existing SCOPE command system we built the Experimental frame, based on DEVS principles for our modular DEVS-NETSIM simulation model, named GENETSCOPE [38]. Figure 12 shows the block architecture of the simulation model. The right-hand box is the system phenomenon that contains the Automatic Link Establishment (ALE), STANAG 5066 protocols used for establishing links and exchanging

3. Response time of a system is defined as the time taken by the system to display significant effect caused by any update in the configuration parameters.
4. A methodology using intermediate XML processing to automate much of the process of "componentizing" legacy simulation code will be reported soon.

data messages between mobile stations and fixed stations. The left-hand box is the experimental frame that generates various scenarios and parameters under study. The scenarios and parameters are fed into the model and performance characteristics are obtained from it, which are then visualized and analyzed in real time as per the extended MVC architecture described in section 5.

8.2 SCOPE Command and DoDAF

Certainly, a system like SCOPE command qualifies to be represented as a DoDAF specification. Though not provided in this paper, all three views, viz., Operational, System, and Technical, can be developed. The documents are fairly easy to construct as the system is not in the design phase but is a live system with working standards and people managing the system for as long as twenty years. The physics of the HF communication is still the same, and the radio equipment has set standards that have not been revised that often. What is new in the system is the incorporation of new standards, for example, the STANAG 5066 data-exchange protocol that modulates the modem rates and reliable data delivery across the HF messaging system. This is being added to provide the capability to send e-mail messages through the HF system. The other major thing that has changed is the increased intensity of traffic, demanding upgrades to the existing system. For illustration purposes, suppose that we had the DoDAF description of SCOPE as well as all the details on how the system would be constructed and its functionality implemented. Remaining solely within the DoDAF, there still would not be any means to analyze or experiment with the projected system. As stressed earlier, the DoDAF does not provide for any M&S capability to support the system design process. It only provides a means to build a system on the presumption that analysis has already been done, a "design" is available, and the system is ready to be deployed. The purpose of the DoDAF in this case is nothing more than a documenting procedure.

The methodology presented in this paper takes the DoDAF as a front-end documentation procedure that aids M&S and design objectives. With respect to Figure 5, the central theme of the paper, we present sample OV-8 and OV-9 documents to illustrate how the experimental frame is developed from the DoDAF terminology.

Although the current DoDAF views are insufficient to provide the M&S for the purpose of enhancing and recommending upgrades to the existing SCOPE system, the DEVS approach readily provides the needed tools. Going back to the basic DEVS M&S components (see Figure 1), the legacy SCOPE simulation model was transformed by the base high resolution model. The Experimental frame is constructed over this existing system along with various other additions that would control and direct the possible upgrades. This component is responsible to provide environmental conditions, workload generation, performance analysis, system evolution and control, and achievement of steady state. The other advantage of this separation is the construction of a DEVS lumped model in which various details of

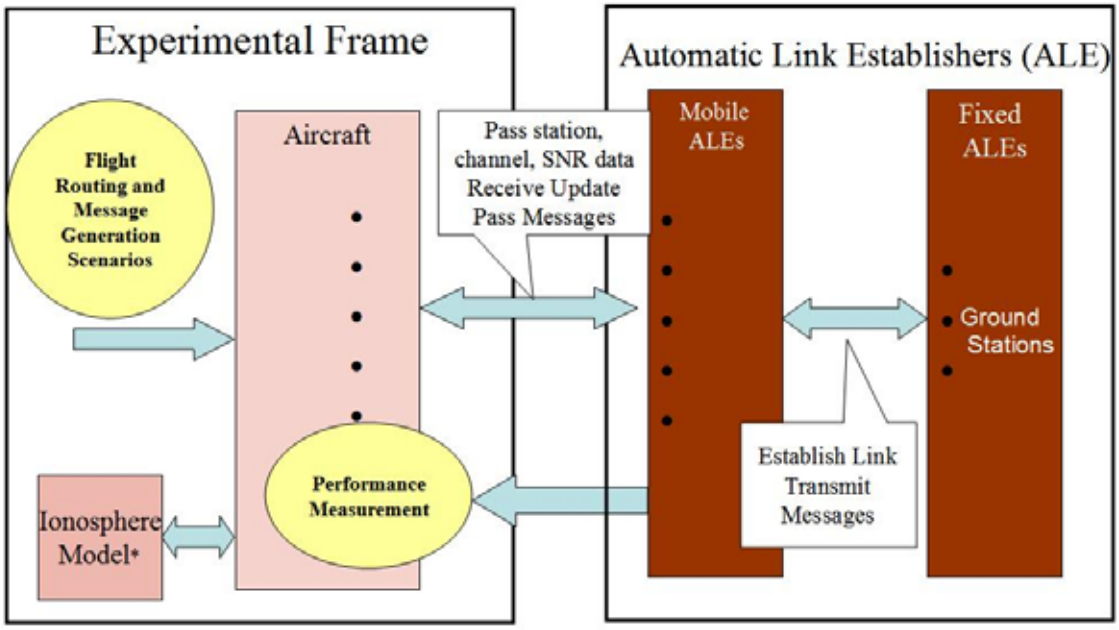

Figure 12. GENETSCOPE simulation architecture for SCOPE command

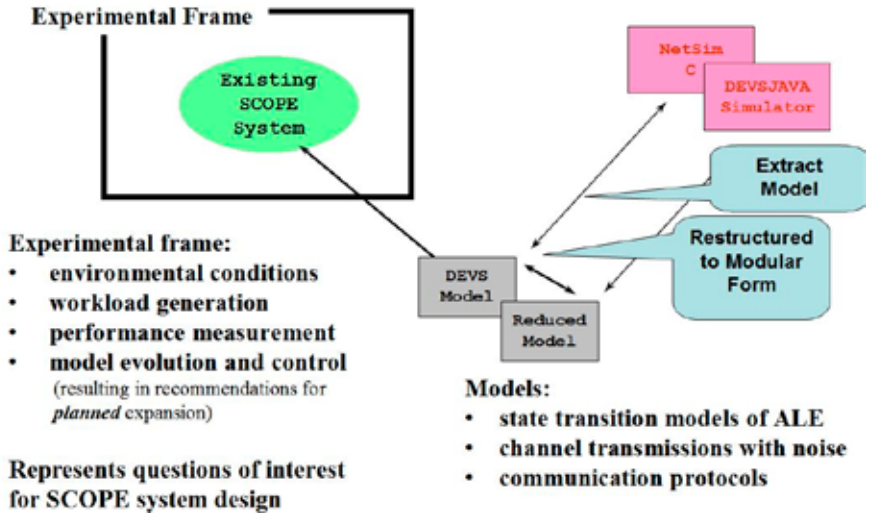

Figure 13. DEVS M&S and the existing SCOPE command system

the base model are abstracted and lumped together. Whereas the base model is oriented to technical components, the lumped model directly addresses system level issues and supports faster simulation runs to answer these questions. As always, the question arises as to how close these results match with the detailed model. The lumped model is preferred if it is able to perform to the same level of accuracy and helps answer the questions raised by the SCOPE command designers. The comparison of a lumped model with a base model is only possible if the underlying M&S formalism supports modular construction of the three components, viz., model, simulator, and Experimental frame [11].

8.2.1 Sample OV-8 and OV-9 Documents

Let's consider two activities out of many activities that are a part of any HF radio communication, i.e., *sounding* and *listening*. Sounding is defined as the process by which different stations (refer to Figure 11) periodically send broadcast messages at different frequencies so that other stations know who else is available on the HF radio sky. Listening is defined as the process by which these stations identify and hear RF tones and go through a demodulation process to decode and decipher the incoming transmission.

Table 4 describes the initial process that is done to populate the OV-8 document. It assigns various IDs to different Activities and sub-activities that are then used as reference tokens and automation processes, as described in [18]. Figure 14 depicts the OV-5 for activity *sounding*. Activity *listening* will have a similar Operational View depiction. Table 5 presents a sample OV-8 document with refined structure (see Table 3) showing the significant parameter set for *sounding* and *listening* activities. It should be well stressed here that documention and aggregation of this information with the corresponding activity helps find faults in testing the "feasibility" of the system [18] when M&S is employed.

Table 4. Activity ID mapping for OV-8 and OV-9

S. No.	Activity	Sub-activity	Internal Activity	ID
1	Sounding			A1
2		Prepare Call		A1.1
3		Send Call		A1.2
4		Send Transmission		A1.3
5	Listening			A2
6		Receive Transmission		A2.1
7		Evaluate Signal		A2.2
8		Decode Signal		A2.3
9		Report Message		A2.4

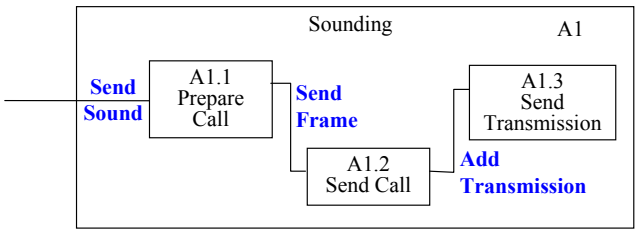

Figure 14. OV-5 for activity sounding

Table 5. Sample OV-8 document

S. No.	Activity ID	Signify-cant Para-meter		Source Activity	Input Interface Name	Message Descript-ion	Container Op. Node	Source Document/ Diagram
1	**A1**						Station	
2	A1.1	**Sounding-interval, duration**	CA1.1	Ax	inSta	Send sound	Station	Figure 14/ OV-5
3	A1.2	Message size, frame count	CA1.2	A1.1	inAle	Send frame (s)	Station	Figure 14/ OV-5
4	A1.3	**Duration**	CA1.3	A1.2	inRt	Add transmission	Station	Figure 14/ OV-5
5	**A2**							
6	A2.1	Duration	CA2.1	Ay	inRt	Receive transmission	Station	Figure x
7	A2.2	Station to-station SNR	CA2.2	A2.1	inAle	SNR	Station	Figure x
8	A2.3	Received frames, valid frames, duration	CA2.3	A2.2	inAle	Incoming sound	Station	Figure x
9	A2.4	None	CA2.4	A2.3	inHfnc	Heard station X	Station	Figure x

Having constructed the OV-8 document, let us construct the OV-9 documents according to the proposed structure in [18]. Table 6 presents the components that lie within the Operational Node *station* and their assigned IDs for automation purposes. For more details, refer to [18]. It is worth stressing here that this information comes readily from the SES of the existing SCOPE command system, as shown in Figure 11. The inner components within the *station* Operational Node are clearly defined in Figure 11.

Hence, during the creation of OV-8 and OV-9 we have populated the model repository with Activity models (MA6.1–MA6.18) and Operational node's inner components models (ME1, ME1.1–ME1.6) and have created an interface between these two aspects of DoDAF design. In the subsequent sections, we shall see how these enhanced OV-8 and OV-9 documents prove to be advantageous in defining the DEVS Experimental frame parameters and hierarchical GUI developments or code development of the simulation model.

8.3 SCOPE Architecture Implementation Using Enhanced MVC

Figure 15 shows the simulation architecture for GENETSCOPE [38] using the concepts laid out in the paper. With reference to Figure 12, the ionosphere model used in the architecture is ICEPAC data. It is worth stressing that the initial NETSIM model written in C language has this database tightly coupled with the model. In our present implementation, we made it modular so that it can be replaced by any other database that could provide the channel propagation values through the ionosphere, e.g., VOACAP. In the current implementation, there is no ICEPAC database included but the complete ICEPAC software that is executed at run time. This is one of the biggest advantages in separating ICEPAC from the model itself. The ICEPAC software is configured through the Experimental frame parameters and is made available for real-time execution as an independent thread for different stations that are active in the running DEVS model. The real-time execution of ICEPAC software involves creation of a dynamic ICEPAC configuration file that contains information about the two stations, their geographical locations in latitude and longitude, the Sun Spot Number (SSN), and the time of year, month, and day. This implementation allows us to get the ionospheric SNR values for any location at any time of the year (for SSN) unlike the earlier implementation (NETSIM-SC) where we were limited to only a handful SSN values (10, 70, 100, and 130) with locations specified in five-degree increments. This has the added benefit of using the exact location of any mobile station rather than using

Table 6. Inner components within operational nodes and their mapping with "standardized" DEVS models

S. No.	Operational Node	Inner Component Entities	Component Name	Associated Models Added to Repository	Hierarchical Parent/ Container	DEVS Model Type
1	O1	OCE1	Station	ME1	-	Digraph
2		OCE1.1	HFNC	ME1.1	ME1	Atomic
3		OCE1.2	ALE	ME1.2	ME1	Atomic
4		OCE1.3	RT	ME1.3	ME1	Atomic
5		OCE1.4	TX	ME1.4	ME1	Atomic
6		OCE1.5	RX	ME1.5	ME1	Atomic
7		OCE1.6	PA	ME1.6	ME1	Atomic

Table 7. Sample OV-9 document

S. No.	Operational Node	Inner Component Entities	Component Name	Activity Component	Activity Component Name	Interface Description
1	O1	OCE1	TCT			
		OCE1.1	HFNC	Ax	Time To Sound	tts
				A2.4	Report Message	repMsg
		OCE1.2	ALE	A1.1	Prepare Call	prepCall
				A1.2	Send Call	sendCall
				A2.2	Evaluate Signal	evalSig
				A2.3	Decode Signal	decSig
		OCE1.3	RT	A1.3	Send Transmission	sendTransm
				A2.1	Receive Transmission	recvTransm
		OCE1.4	TX	A1.3	Send Transmission	putTransm
		OCE1.5	RX	A2.1	Receive Transmission	getTransm
		OCE1.6	PA	None	None	None

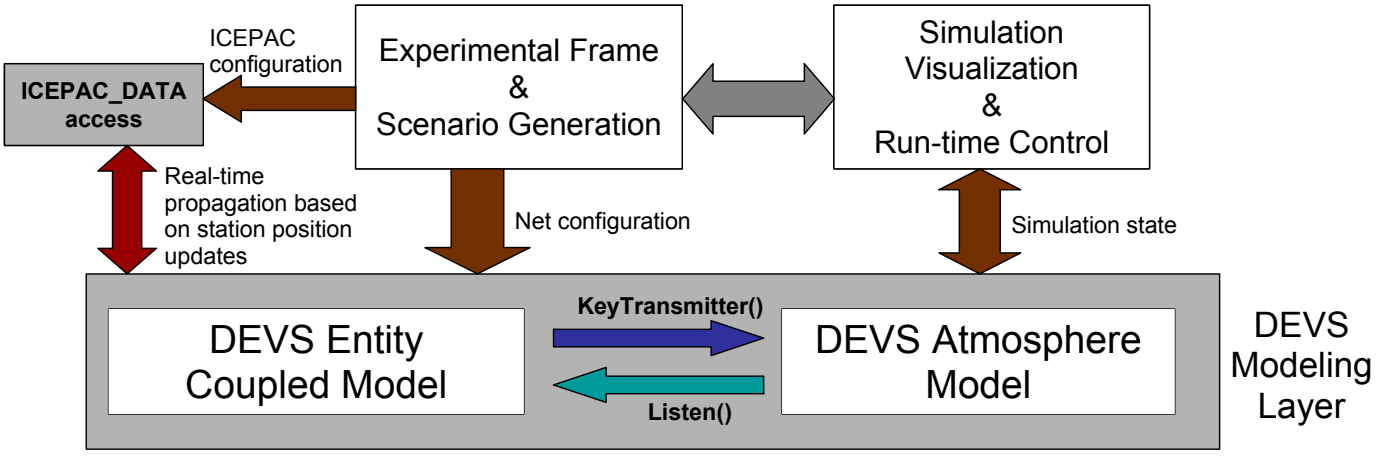

Figure 15. Simulation architecture for the SCOPE command network

projections within the implemented grid as in the earlier NETSIM-SC. The DEVS layer comprises both models as well as the DEVS simulation environment. The Experimental frame layer also contains the controls required to modify/update the model as well as a simulator as per enhanced MVC. The simulation visualization is modular in construction and reflects the updates in the Experimental frame layer and the DEVS layer. See Figure 15.

The above architecture is shown below in various screen shots taken from the developed GENETSCOPE (beta version). Figure 16 shows the Experimental frame and various parameters (along with their default values) used in scenario configuration. The parameters shown in bold below are the parameters that have been identified as *significant* parameters in OV-8 (see Table 5, in shaded cells). Similarly, other parameters too come from an elaborate OV-8 document of the SCOPE command. These significant parameters find their way in various configurable parameters all through the model configuration settings as shown in Figures 16, 17, 18, 20, and 21, and the simulation model finds its design through the SES (see Figure 11) or the corresponding OV-9 document (see Table 6). The total parameter set is comprised of:

1) Number of fixed stations,
2) Number of levels inside a fixed station,
3) Number of mobile stations (aircrafts),
4) Messages per hour,
5) **Data message size,**
6) **Voice call duration,**
7) **Ground stations sounding interval, and**
8) SNR threshold for a received signal.

Once the experimental frame parameters are configured, these parameters are channeled down to the individual components. The top-level design parameters then bound the other internal component parametric settings. For example, Figure 17 shows a typical configuration of the ground station Sigonella. The left column in Figure 17 shows all the fourteen ground stations, and the individual details about each station can be seen by pressing the *Lookup* button. Figure 17 also shows the message traffic that is transmitted by this station. Notice that the Experimental frame settings are shown as the traffic stream originated from this station. Similarly, a mobile station configuration panel is shown in Figure 18. The user can select any specific mobile aircrafts bounded by the number of mobile stations specified in the Experimental frame. The next figure, 19, basically lets the user enter call-signs to these mobile stations and invites the user to enter aircraft-specific details like message traffic, flight

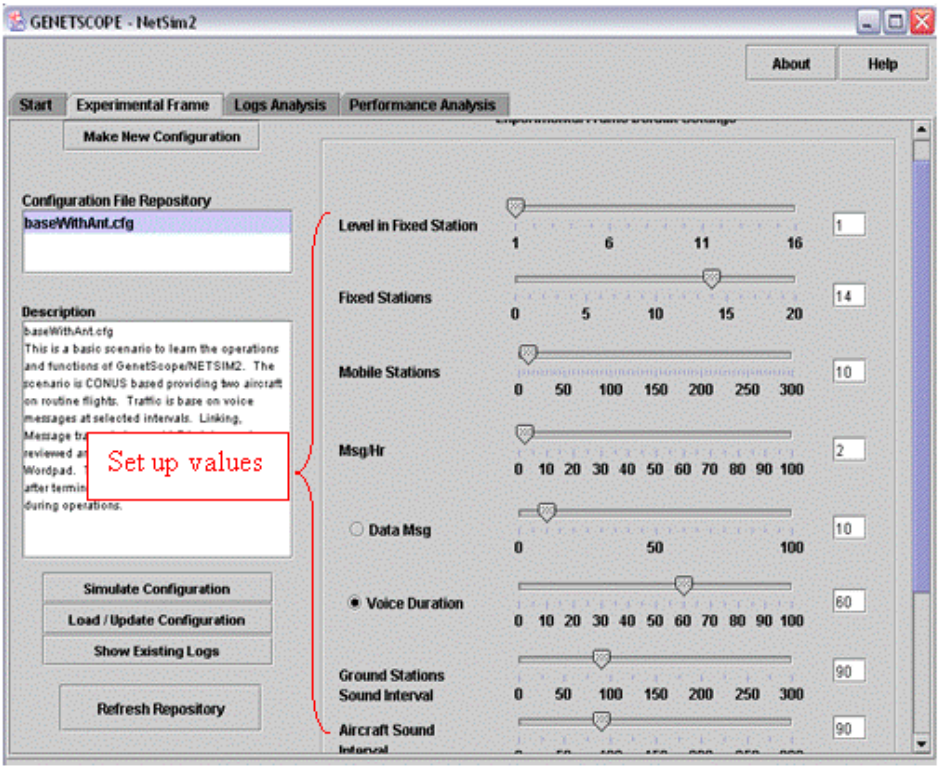

Figure 16. Experimental frame for GENETSCOPE

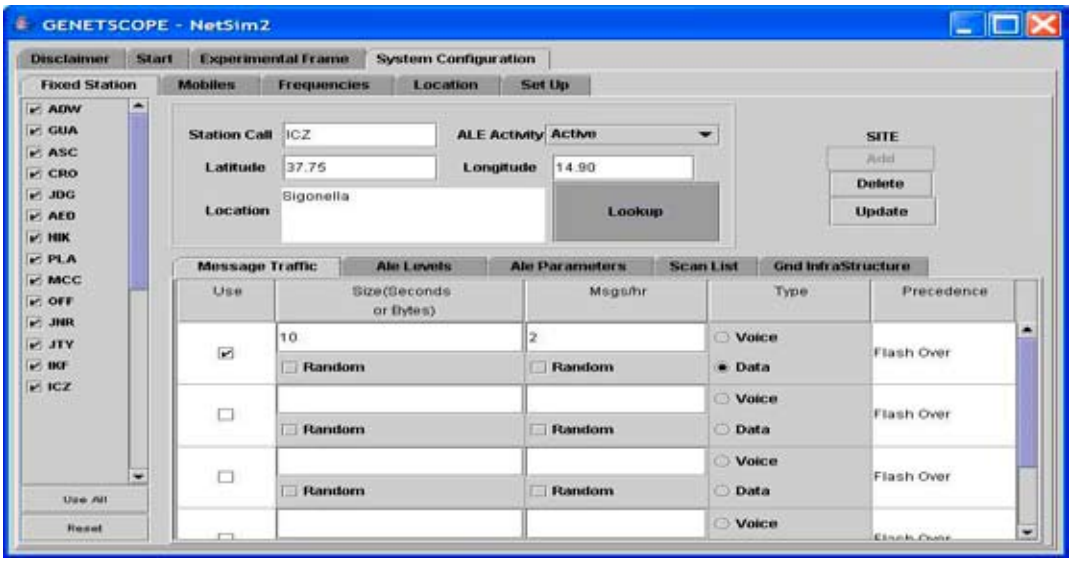

Figure 17. Ground station configuration screen for Naval Air Station Sigonella

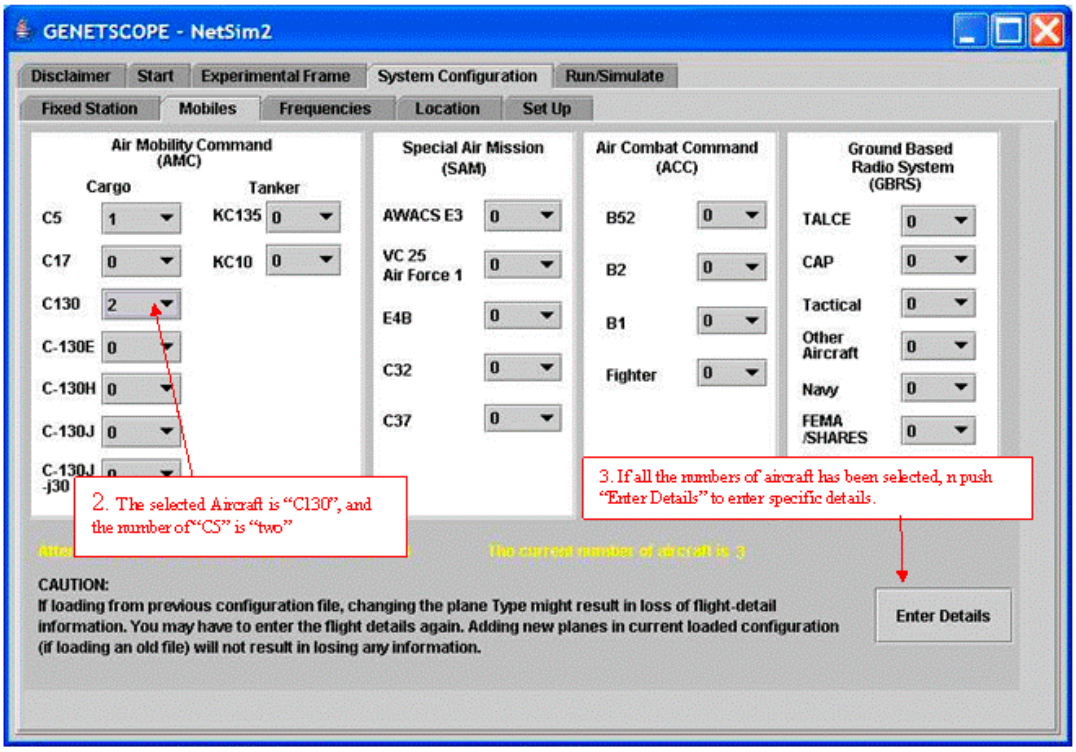

Figure 18. Mobile station configuration screen where the total count is bounded by the Experimental frame

Figure 19. Callsign entry for a mobile station

Figure 20. Flight path of mobile aircraft and other details

path (see Figure 20), radio parameters, and channel frequencies being used. Other internal details of station configuration can be seen in the GENETSCOPE software user's manual [38]. The purpose of showing GUI snapshots in Figures 17–20 is to illustrate how top-down design parameters (from OV-8) can be taken down to the component level (through both OV-8 and OV-9). The other important aspect of this process is that during simulation run-time, if the Experimental frame parameters are changed to study any particular parameter, that change is channeled across the whole system model configuration using "interrupts," thereby exploiting the discrete event simulation methodology. The update of any Experimental frame parameter is taken by the simulation model as an "external" event.

The last piece of information being fed through the Experimental frame is the ICEPAC setting, based on the Sun Spot Number (SSN). Once the system model is configured through the Experimental frame settings, the user is directed toward the simulation setup. Figure 21 shows the final setup screen after which the user then moves on to the run-time simulation screen (see Figure 22) to execute the simulation. When the user clicks the Write Files button in Figure 20, it results in writing up of the detailed configuration file for repository purposes.

Figure 22 shows the simulation clock as it happens in real time and the obtained statistics. The above snapshots complete the architectural components specified in Figure 15. Figure 22 has the functionalities that are described earlier in the paper: e.g., run-time configuration updating and simulation control. It has four buttons at the top of the screen, viz.:

1) *Run Abstract Model* (using lumped parameters),
2) *Run Detailed Model* (using detailed parametric settings),
3) *Pause* (to interrupt the simulation),
4) *Terminate* (to end the simulation).

The *Pause* button is of special interest here, as the user can interrupt the running simulation (manual reactive control described in section 7.3.2) and change the Experimental frame or system configuration settings while the simulation is in action. Once the parameters have been updated, the user can resume the simulation and can see the impact of that update on the above "active" simulation visualization screen. One such example may be the two obtained values of total transmissions and total sounds heard. If the number of sounds heard is not up to the mark (with respect to a validated real-world scenario), the user may change sound-interval time or any other parameter that would impact this number, or may conclude that the model is not "performing" correctly. The rapid impact of any such parameter can be studied by pausing the simulation and changing it and then observing the effects in the simulation pane.

The DEVS layer in Figure 15 is implemented in the following manner. The simulation engine running behind uses the following code.

```
NetsimSC net = new NetsimSC
(createdConfigFile, debugOption);
tCoord = new TunableCoordinator(net);
tCoord.initialize();
tCoord.setTimeScale(0.0001);
tCoord.simulate(Integer.MAX_VALUE);
```

The model configuration is written into a configuration file that is used to create the DEVS digraph model, with automated coupling using the system SES shown in Figure 11. The DEVS model is then passed on to the TunableCoordinator derived from DEVS RTcoordinator class. The TunableCoordinator is

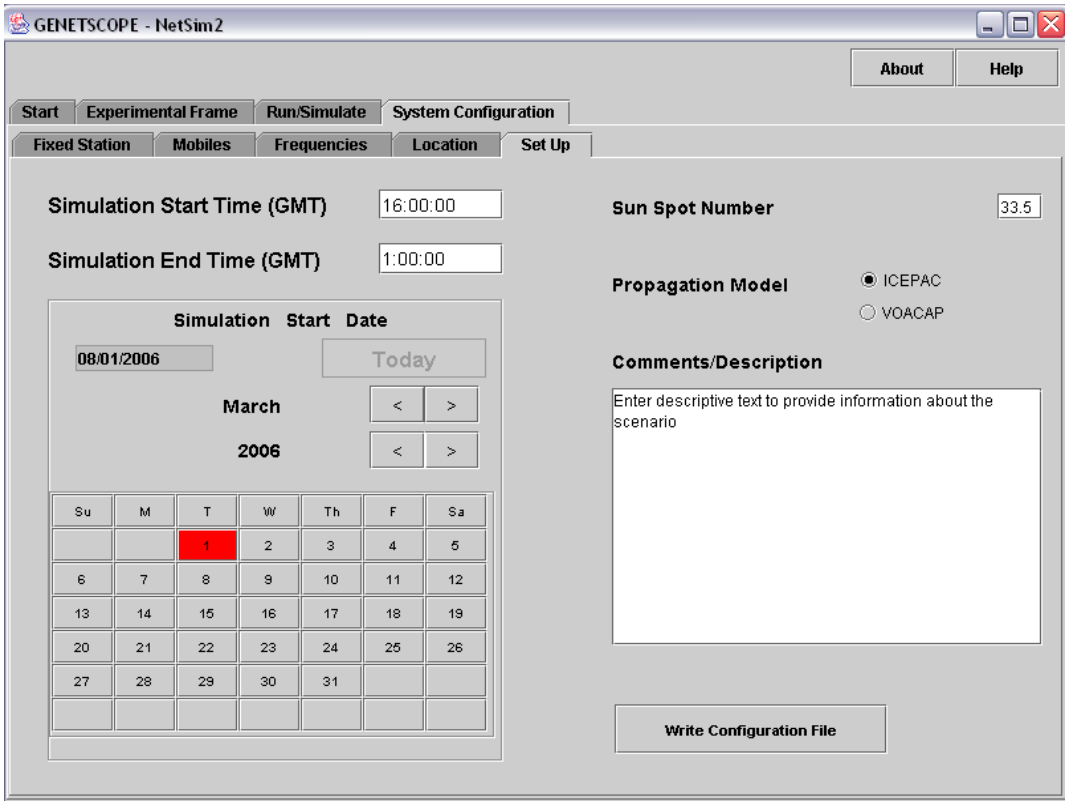

Figure 21. Experimental frame and ICEPAC data configuration through selection of SSN

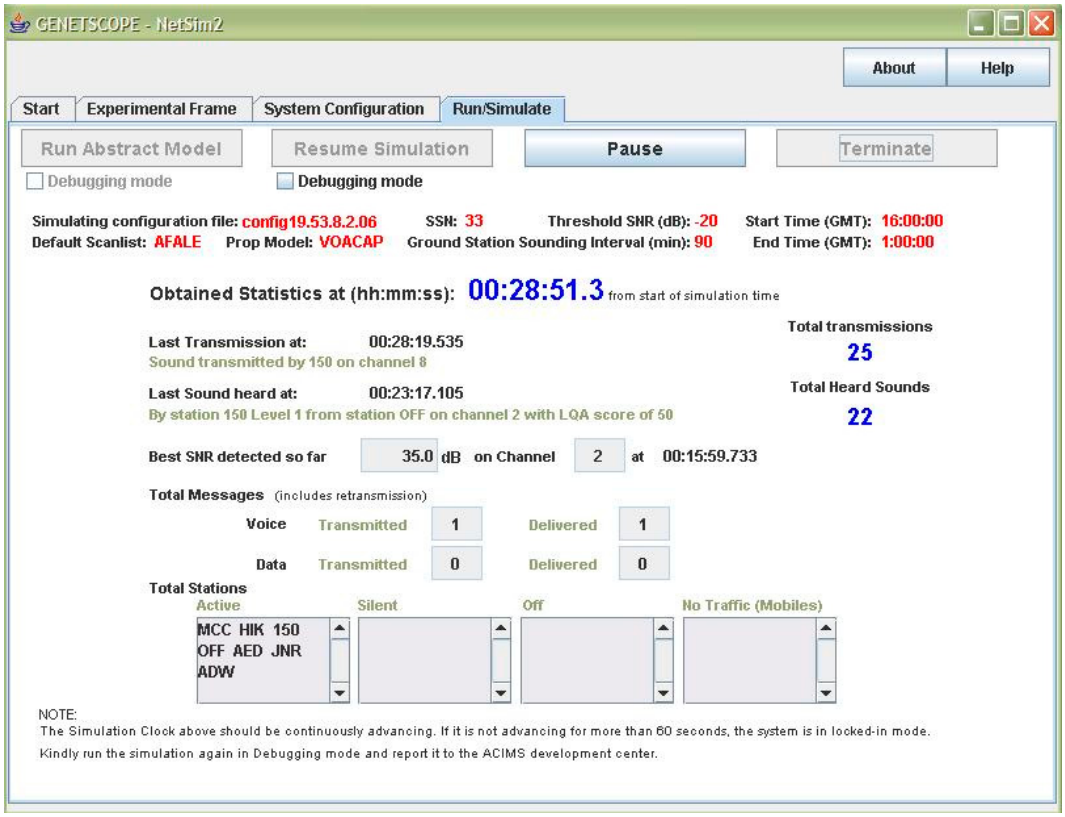

Figure 22. Run-time simulation visualization screen for rapid feedback

initialized and is then directed to simulate for a maximum number of iterations, which means that simulation will proceed indefinitely (in logical sense). The *Pause* button executes the following line.

```
tCoord.interrupt();
```

After the simulation is paused and updates are made, the simulation is restarted by simply calling the coordinator to "simulate."

```
tCoord.simulate(Integer.MAX_VALUE);
```

The simulation core functionality provided by the DEVS simulation protocol facilitates interrupting the coordinator and makes real-time parametric and component structures at run time as described in sections 6 and 7 earlier.

Figure 22 contains a very limited set of aggregated information. However, run-time graphs and projections can be very well aligned with this visualization to see patterns and the direction in which the simulation is proceeding. Logs are generated for each simulation run. This visualization pane shows the important information of the Experimental frame (in red) and the run-time information from the system model (in blue), which, needless to say, is according to the enhanced MVC (through the development of appropriate interfaces between these layers). The View layer (see Figure 5) in the current example shows only the model and the Experimental frame control visualization. The Experimental frame control is *controller B* in Figure 5, i.e., parameters that "control" the model. The lowest layer, i.e., *controller A* in the enhanced MVC process, is not the focus of the GENETSCOPE project and consequently not illustrated here. Its implementation is illustrated in our earlier work [28].

8.4 Implications of the Example Above and NR-KPP

Having laid out the framework to conduct and design the experiments, the next item on the agenda is to identify the *measures of effectiveness* (MoEs) that eventually will be considered in making recommendations for any update or modification needed in the current SCOPE command infrastructure. Since the SCOPE command is a deployed system, we were given various statistical reports by JITC [36] in order to determine these MoEs. The point of this exercise is to provide sufficient analysis through simulation of the modeled system so that the impact of any particular infrastructural change intended in the system can be observed on these MoEs. Some of the MoEs that were identified are as follows:

1) Longest time taken by any e-mail on HF network,
2) Number of e-mails sent and number of e-mails actually delivered,
3) Average message transmission time at any station per hour,
4) Messages attempted versus messages received per hour,
5) Bandwidth usage at Central Network Command Station (CNCS[5]),
6) Number of planes in "good" signal to noise ratio (SNR) range per hour.

The parameters that are to be set in order to recommend any upgrades in the current infrastructure can be listed as follows:

1) Average number of daily flights,
2) Minimum number of messages attempted by any station,
3) Number of fixed stations participating in any mission scenario,
4) Number of active levels within a fixed station,
5) Minimum and maximum message size in KB,
6) Minimum and maximum duration of a phone call (VOICE message),
7) Minimum data rate by any ALE radio-modem.

As can been seen clearly, there is not a one-to-one mapping between MoEs and experimentation parameters. The MoEs tell us about the effectiveness of any mission that would be executed. They are holistic measures that tell about the fitness, capacities, and limitations of the system. M&S is the preferred means for assessing the impact of parameters on MoEs, with the goal of determining the most significant parameters. A simulation execution environment can help this investigation through a rapid feedback cycle where the analyst can change parameter values on the fly and quickly assess their impact on holistic measures. These MoEs impact evaluations very well and become part of the result set as mentioned in section 7.4.2, while the parameters identified become part of the Experimental frame layer as shown in Figure 15.

Similarly, for any DoDAF architecture, the MoEs are also specialized for that particular architecture. Considering the breadth of the SCOPE command system, some of the MoEs mentioned above also apply to any net-centric architecture. Within the DoD, JITC has the sole responsibility of certifying the Information Technology (IT) and National Security Systems (NSS) for interoperability purposes [37]. The major T&E problem identified today by JITC is how

5. CNCS is the gateway for any land-based network (SIPRNET or NIPRNET) to be connected to the SCOPE command HF network. All e-mails are routed through CNCS.

to verify that a solution provided by any architecture is data integrated and net centric in operation. The traditional T&E approaches are optimized to verify performance and effectiveness of point solutions, but new criteria are needed to reflect the realities of systems operating within networked systems. Such criteria are just beginning to emerge and are not yet matured for immediate and widespread use of T&E [37].

The NR-KPP assesses net-readiness information assurance (IA) requirements, and end-to-end operational effectiveness of that exchange with respect to the COIs mentioned earlier. Description of Key Interface Profile (KIP) with relation to this scenario is beyond the scope of this paper. The major object underlying NR-KPPs is to identify verifiable performance parameters and associated metrics required to evaluate timely, accurate, and complete exchange and use of information to satisfy the information needs for a given capability [37].

9. Conclusions

This paper has provided a contribution in proposing an enhanced MVC framework that aids the DEVS modeling and simulation framework. The enhanced MVC complements the basic DEVS framework components, viz., the Experimental frame, the model, and the simulator. The integration of these two frameworks results in a well constructed control panel that provides a more comprehensive feature set and controls to calibrate the model and configure the simulation. The recent advances in DEVS technology, like variable structure modeling, real-time simulation tuning with rapid feedback, and model/simulator calibration, have been described; they help in the analysis and study of fast-changing network scenarios. The first major advantage of incorporating these technologies is the study and visualization of the "transition" effects when the model configuration is modified in a running simulation. Various methods of controlling simulation execution were explored as well as ways in which they can be used in different scenarios. The second major advantage of this enhanced MVC framework is the capability to reach the desired mission effectiveness or performance benchmarks in an active simulation. With variable structure capability, along with setting the bounds of any result parameter, the system can be observed to arrive at the corresponding "steady state." This methodology also aids in determining the most significant parameters for any complex system for which theoretical analysis is not feasible. This integrated framework is applied to the enhanced DoDAF document that comprises two new operational views, OV-8 and OV-9, which are dedicated to the M&S areas. These two documents are refined to incorporate the control parameter set for the control panel, specified as a significant parameter set in OV-8 and OV-9. These parameters are discussed with respect to the Net-Ready Key Performance Parameter (NR-KPP) set, and the advantages of identification of these parameters during the operational view design phase are emphasized. Finally, a working example for the Systems Capable of Planned Expansion (SCOPE) command system is provided to illustrate the concepts and DEVS capabilities.

10. References

[1] Carstairs, D.J. "Wanted: A New Test Approach for Military Net-Centric Operations," Guest Editorial. *ITEA Journal* 26(3, Oct 2005).

[2] Hu, X, and B.P. Zeigler. "Model Continuity in the Design of Dynamic Distributed Real-Time Systems." In *IEEE Transactions on Systems, Man and Cybernetics— Part A: Systems and Humans,* 2006.

[3] Wegmann, A. "Strengthening MDA by Drawing from the Living Systems Theory." Workshop in Software Model Engineering, 2002.

[4] DoD Architecture Framework. *Software Productivity Consortium.* (cited January 9, 2005). Available from: http://www.software.org/pub/architecture/dodaf.asp

[5] DoD Instruction 5000.2. *Operation of the Defense Acquisition System.* May 12, 2003.

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

[7] Chairman, JCS Instruction 6212.01D. *Interoperability and Supportability of Information Technology and National Security Systems.* March 6, 2006. Avaible from: http://jitc.fhu.disa.mil/jitc_dri/pdfs/6212_01.pdf

[8] Atkinson, K. "Modeling and Simulation Foundation for Capabilities Based Planning." Simulation Interoperability Workshop, Spring 2004.

[9] Zeigler, B.P., and S. Mittal. "Enhancing DoDAF with DEVS-Based System Life-Cycle Process." IEEE International Conference on Systems, Man and Cybernetics, Hawaii, October 2005.

[10] ACIMS Software Development Website. [cited September 2006]. Available from: http://www.acims.arizona.edu/SOFWARE/software.shtml

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

[12] Buss, A., and L. Jackson. "Distributed Simulation Modeling: A Comparison of CORBA, HLA, and RMI." In *Proceedings of the 1998 Winter Simulation Conference,* 1998.

[13] Sarjoughian, H.S., and F.E. Cellier, eds. *Discrete Event Modeling and Simulation Technologies: A Tapestry of Systems and AI-Based Theories and Methodologies.* New York: Spring-Verlag, 2001.

[14] Tolk, A., and S. Solick. "Using the C4ISR Architecture Framework as a Tool to Facilitate V&V for Simulation Systems Within the Military Application Domain." Simulation Interoperability Workshop, Spring 2003.

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

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

[17] Cho, Y., B.P. Zeigler, and H.S. Sarjoughian. "Design and Implementation of Distributed Real-Time DEVS/CORBA." IEEE Sys. Man. Cyber. Conference, Tucson, AZ, October 2001.

[18] Mittal, S. "Extending DoDAF to Allow Integrated DEVS-Based Modeling and Simulation." Special Issue on DoDAF, *Journal of Defense Modeling and Simulation* 3(2, April 2006).

[19] Gaetjen, T. "Net-Ready Key Performance Parameters Testing" (cited October 2005). Available from: www.opengroup.org/gesforum/uploads/40/4548/NR_KPP_Testing_v1.pdf

[20] Krasner, G., and S.A. Pope. "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System." *Journal of Object Oriented Programming* 1(3, 1988): 26–49.

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

[22] Mittal, S., B.P. Zeigler, P. Hammonds, and M. Veena. "Network Simulation Environment for Evaluating and Benchmarking HLA/RTI Experiments." JITC Report, Fort Huachuca, December 2004.

[23] Hu, X., B.P. Zeigler, and S. Mittal. "Dynamic Configuration in DEVS Component-Based Modeling and Simulation." *SIMULATION* (November 2003).

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

[25] Zeigler, B.P., D. Fulton, P. Hammonds, and J. Nutaro. "Framework for M&S-Based System Development and Testing in Net-Centric Environment." *ITEA Journal* 26(3, 2005).

[26] DoDAF Working Group. *DoD Architecture Framework Version 1.0, Vol III: Deskbook.* DoD, August 2003.

[27] Barr, B., R.D. Aaron, D.T. Hill, and P.H. Christensen. "Net Ready Key Performance Parameter (NR-KPP) Test and Evaluation Strategy." *ITEA Journal* 26(3, 2005).

[28] Nutaro, J., and P. Hammonds. "Combining the Model/View/Control Design Pattern with the DEVS Formalism to Achieve Rigor and Reusability in Distributed Simulation." *Journal of Defense Modeling and Simulation* 2(May 2005).

[29] Brown, A.W., and K.C. Wallnau. "The Current State of CBSE." *IEEE Software* 15(5, 1998): 37–46.

[30] Zeigler, B.P., and Reynolds, R.G. "Towards a theory of adaptive computer architectures." In *Proceedings of the 5th International Conference on Distributed Computing Systems,* 1985, 468–475.

[31] Zeigler, B.P., and A. Louri. "A Simulation Environment for Intelligent Machine Architecture." *Journal of Parallel and Distributed Computing* 18 (1993): 77–88.

[32] Chean, M., and L.A.B. Fortes. "A Taxonomy of Reconfigurable Techniques for Fault-Tolerant Processor Arrays." *IEEE Computer* 23(1, 1990): 55–69.

[33] Uhrmacher, A.M. "Variable Structure Models: Autonomy and Control—Answers from Two Different Modeling Approaches." In *Proceedings on AI, Simulation, and Planning in High Autonomy Systems,* 1993, 133–139.

[34] Chen, X. "Dependence Management for Dynamic Reconfiguration of Component-Based Distributed Systems." In *Proceedings of the 17th IEEE International Conference on Automated Software Engineering,* 2002, 279–84.

[35] Department of Defense. *Department of Defense Interoperability Communications Exercise (DICE).* 2005. (cited October 2005). Available from: jitc.fhu.disa.mil/dice_conf/2005/initial/downloads.

[36] JITC. *JITC Reports for SCOPE Command for Year 2005.* (latest data as of October 2005).

[37] Buchheister, J.B. "Net-Centric Test & Evaluation. Command and Control Research and Technology Symposium: The Power of Information Age Concepts and Technologies," 2004. (cited October 2005). Available from: www.dodccrp.org/events/2004/CCRTS_San_Diego/CD/papers/2008.pdf

[38] GENETSCOPE (Beta Version) Software User's Manual. Available from: ACIMS Center, University of Arizona.

[39] Kim, T.G., C. Lee, E.R. Christensen, and B.P. Zeigler. "System Entity Structuring and Model Base Management." *IEEE Transactions on Systems, Man and Cybernetics* 20(5, 1990).

Appendix: System Entity Structure (SES)

The SES formalism is a structural knowledge representation scheme that systematically organizes a family of possible structures of a system. Such a family characterizes decomposition, coupling, and taxonomic relationships among entities. An *entity* represents a real-world object. The decomposition of any entity concerns how it may be broken down into subentities. Coupling specifications tell us how different subentities can be coupled together to reconstitute an entity. The taxonomic relationship concerns admissible variants of an entity. It also provides a formal framework for representing the family of possible structures. From a design point of view, SES represents the design space with various possible design configurations. Thus, the process of design/analysis is to prune SES—in other words, to search the best design configuration. For complex systems, the number of the combination of different configurations is very large. Thus, it is desirable to be able to emulate SES and automatically search the best design configuration. For a detailed description on SES see [11] and [39].

Author Biographies

Saurabh Mittal *is a research engineer at Arizona Center of Modeling & Simulation (ACIMS) at University of Arizona. He is also a Ph.D. candidate in Electrical & Computer Engineering (ECE) at the University of Arizona. He is a recipient of JITC's highest civilian contractor award 'Golden Eagle' for the project GENETSCOPE. He holds an M.S in ECE from the University of Arizona. His research interests include DEVS based integrated design methodologies, DoDAF, modeling languages, automated testing and design of software systems. He can be reached at saumitt@gmail.com.*

Eddie Mak *is the lead software developer of ATC-Gen at Joint Interoperability Test Command (JITC). He is a principal system programmer at Arizona Center for Integrative Modeling and Simulation (ACIMS) in Tucson, Arizona.* Eddie *holds an M.S. in Electrical and Computer Engineering from the University of Arizona. He can be reached at eddie.mak@gmail.com*

James Nutaro *received his PhD from the University of Arizona in 2003. He is currently a part of the research staff at Oak Ridge National Laboratory in the Computation Sciences and Engineering Division. His research interests include formal methods for discrete event systems and discrete event simulation, hybrid system modeling, and model based system design. He can be reached at nutarojj@ornl.gov*