DEVS Unified Process for Web-Centric Development and Testing of System of Systems

Saurabh Mittal, Bernard P. Zeigler
Arizona Center for Integrative Modeling and Simulation,
ECE Dept. University of Arizona, Tucson, 85721
{saurabh|zeigler}@ece.arizona.edu

Abstract

A critical aspect and differentiator of a System of Systems (SoS) versus a single monolithic system is interoperability among the constituent disparate systems. A major application of Modeling and Simulation (M&S) to SoS Engineering is to facilitate system integration in a manner that helps to cope with such interoperability problems. A case in point is the integration infrastructure offered by the DoD Global Information Grid (GIG) and its Service Oriented Architecture (SOA). In this paper, we discuss a process called DEVS Unified Process (DUNIP) that uses the Discrete Event System Specification (DEVS) formalism as a basis for integrated system engineering and testing called the Bifurcated Model-Continuity life-cycle development methodology. DUNIP uses an XML-based DEVS Modeling Language (DEVSML) framework that provides the capability to compose models that may be expressed in a variety of DEVS implementation languages. The models are deployable for remote and distributed real-time executing agents over the Service Oriented Architecture (SOA) middleware. We also compare DUNIP with the Model Driven Architecture (MDA) paradigm and provide overview of various projects that led to the formulation of DUNIP.

1. Introduction

In an editorial [Car05], 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 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 [Hux04] and model-driven design (MDD) development [Weg02], making test and evaluation an integral part of the design and operations life-cycle.

The performance and acceptance of any software system depends on the validation by the customer that is in part supported by the quality of the test-suite that conducts tests on it. Consequently, it also depends on the quality of the test cases used during the validation process.

Model-based Software Engineering process is commonly referred as Model Drive Architecture (MDA) or Model-Driven Engineering or MDD. The basic idea behind this approach is to develop model before the actual artifact or product is designed and then transform the model itself to the actual product. The MDA is pushed forward by Object Management Group (OMG) since 2001. The MDA approach defines system functionality using platform-independent model (PIM) using an appropriate domain-specific language. Despite such positive benefits of MDA, it lacks sufficient foundation needed to realize this vision. It is underpinned by a variety of standards, some of which have to specified (e.g. executable UML). It is too idealistic and doesn't involve round-trip iterative nature of software engineering and systems engineering perspective. CORBA also pushed forward by OMG failed to provide distributed collaborative environment and execution.

DEVS formalism [Zei00] exists in many implementations, primarily in DEVS/C++ and

DEVSJAVA [ACI06]. Extensions of these implementations are available as DEVS/HLA [Sar01], DEVS/CORBA [Cho01], cell-DEVS [Wai01], and DEVS/RMI [Zha05]. Since DEVS is inherently based on object oriented methodology, and categorically separates the model, the Simulator and the Experimental frame. However, one of the major problems in this kind of mutually exclusively system is that the formalism implementation is itself limited by the underlying programming language. In other words, the model and the simulator exist in the same programming language. Consequently, legacy models as well as models that are available in one implementation are hard to translate from one language to another even though both the implementations are object oriented. Other constraints like libraries inherent in C++ and Java are another source of bottleneck that prevents such interoperability.

In this research effort we propose a new process called DEVS Unified Process (DUNIP) [MIT07g] that utilized the Bifurcated Model-Continuity based life-cycle methodology for a model-based design, execution and collaboration for DEVS models. The life-cycle begins by specifying the system requirements in structured and restricted English that facilitate the requirements gathering from the user. Further, methodologies have been developed to generate DEVS models from BPMN/BPEL-based and message-based requirement specifications and various other input formats. The DEVS models are auto-generated from the specifications and are made available for distributed collaboration using the DEVS Modeling Language (DEVSML) framework. The motivation for this work stems from this need of model interoperability between the disparate simulator implementations and provides a means to make the simulator transparent to model execution. Service Oriented Architecture (SOA) provides the needed framework to develop interoperable applications. In our case it is the interoperable modeling and simulation framework. Although the performance of SOA for a particular simulation exercise is in work, the basic framework of model interoperability has been achieved. Platform Independent Models (PIM) that are based on XML have been implemented in Platform Specific Implementations (PSM) such as Java and .Net [MIT07h]. The interoperability has to exist at both the modeling and the simulation layers. The models are made available for remote and distributed execution using the SOA framework through our developed DEVS/SOA simulation architecture [MIT07c]. The central point resides in executing the simulator as a web service, a capability that is only provided by SOA-based framework. Having a DEVS/SOA simulation framework allows us to simulate models that communicate in a platform independent manner, i.e. the messages are exchanged using XML. The development of this kind of frameworks will help to solve large-scale problems and guarantees interoperability among different networked systems and specifically DEVS-validated models.

In our earlier work, we have also proposed a mapping of DoDAF architectures into a computational environment that incorporates dynamical systems theory and a modeling and simulation (M&S) framework [Mit06a]. The methodology will support complex information systems specification and evaluation using advanced simulation capabilities. Specifically, the DEVS formalism will provide the basis for the computational environment with the systems theory and M&S attributes necessary for design modeling and evaluation. We have demonstrated how this information is added and harnessed from the available DoDAF products towards development of an extended DoDAF integrated architecture that is “Executable”. There are potential advantages of making DoDAF, a DEVS compliant system. We explore the problem of DoDAF using our developed DUNIP framework.

We also enumerate applications of DUNIP in many active and ongoing research projects. To name a few: the GENETSCOPE project [Gen06] and the ATC-Gen project [Mak07] are in current use at Joint Interoperability Test Command (JITC).

2. Background

2.1 Model-Based Software Engineering Process

The basic idea behind this approach is to develop model before the actual artifact or product is designed and then transform the model itself to the actual product. The MDA is pushed forward by Object Management Group (OMG) since 2001. The MDA approach defines system functionality using platform-independent model (PIM) using an appropriate domain-specific language. Then given a Platform Definition Model (PDM), the PIM is translated to one or more platform-specific models (PSMs). The OMG documents the overall process in a document called MDA guide. MDA is a collection of various standards like the Unified Modeling Language (UML), the Meta-Object Facility (MOF), the XML Metadata Interchange (XMI), Common Warehouse Model (CWM) and a couple of others. OMG focuses Model-driven architecture on forward engineering i.e. producing code from abstract, human-elaborated specifications.

It is not required that one tool may contain all of the features needed for Model Driven Engineering. UML is a small subset of much broader scope of UML. Being a subset of MDA, the UML is bounded by its own UML metamodel. Progress has been made to develop executable UML models but it has not gained industry wide mainstream acceptance for the same limited scope.

2.2 Model-Based Testing

Model-based Testing is a variant of testing that relies on explicit behavior models that encode the intended behavior of the system and possibly the behavior of its environment [Utt06]. Pairs of input and output of the model of the implementation are interpreted as test-cases for this implementation: the output of the model is the expected output of the system under test (SUT). This testing methodology must take into account the involved abstractions and the design issues that deals with lumping different aspects as these can not be tested individually using the developed model.

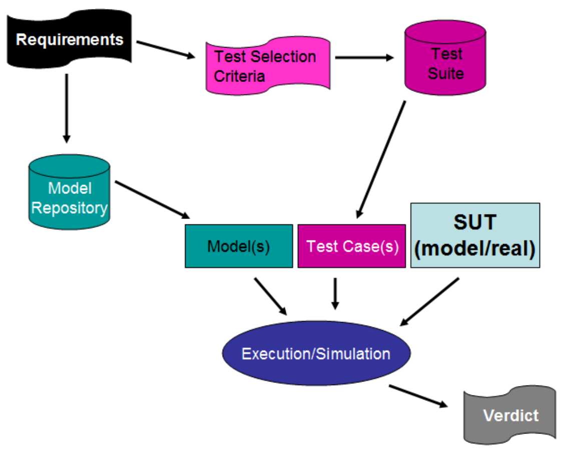


Figure 1: Graphical process extended further from [Utt06]

Following is the process for Model-based testing technique [Utt06] as shown in Figure 1:

1. a model of the SUT is built on existing requirements specification with desired abstraction levels
2. Test selection criteria are defined with an objective to detect severe and likely faults at an acceptable cost. These criteria informally describe the guidelines for a test suite.
3. Test selection criteria are then translated into test case specifications. It is an activity where a textual document is turned 'operational'. Automatic test case generators fall into this step of execution.
4. A test suite is 'generated' that is built upon the underlying model and test case specifications.
5. Test cases from the generated test suite are run on the SUT after suitable prioritization and selection mechanism. Each run results in a verdict of 'passed' or 'failed' or 'inconclusive'.

A summary of contributions to the Model-based Testing domain can be seen at [Utt06].

2.3 DEVS Modeling and Simulation Framework

Discrete Event System Specification (DEVS) [ZEI00] is a formalism, which provides a means of specifying the components of a system in a discrete event simulation. In DEVS formalism, one must specify *Basic Models* and how these models are connected together. These basic models are called *Atomic Models* and larger models which are obtained by connecting these atomic blocks in meaningful fashion are called *Coupled Models* (Figure 2). Each of these atomic models has *inports* (to receive external events), *outports* (to send events), set of *state variables*, *internal transition*, *external transition*, and *time advance functions*. Mathematically it is represented as 7-tuple system: $M = < X, S, Y, \delta_{\text{int}}, \delta_{ext}, \lambda, t_a >$ where X is an input set, S is set of states, Y is set of outputs, δ_{int} is internal transition function, δ_{ext} is external transition function, λ is the output function, and t_a is the time advance function. The model's description (implementation) uses (or discards) the message in the event to do the computation and delivers an output message on the outport and makes a state transition. A Java-based implementation of DEVS formalism, DEVSJAVA [SAR00]**,** can be used to implement these atomic or coupled models. In addition, DEVS-HLA [SAR00] will be helpful in distributed simulation for simulating multiple heterogeneous systems in the System of systems framework.

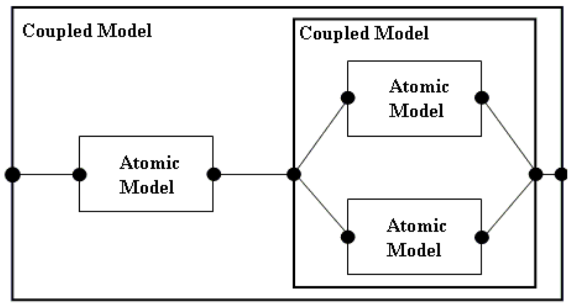


Figure 2: DEVS atomic and coupled models

3. DUNIP Elements

The development of distributed testing environment as advocated by Carstairs 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 [Dod03a, Dod03b, CJC04, CJC06]. Unfortunately, DoDAF and DoD net-centric [Atk04] 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. Combining the systems theory, M&S framework and model-continuity concepts leads naturally to a formulation of a Bifurcated Model-Continuity based Life-cycle process as illustrated in Figure 3.

The process can be applied to development of systems using model-based design principles from scratch or as a process of reverse engineering in which requirements have already been developed in an informal manner. As we shall see ahead in later sections, the said process is used in both manners. The depicted process is a universal process and is applicable in multiple domains. The objective of this research effort is to incorporate DEVS as the binding factor at all phases of this universal process.

Bifurcated Model-Continuity Based Life-Cycle Methodology

The process has the following characteristics [ZEI05]:

- **Behavior Requirements at lower levels of System Specification**: The hierarchy of system specification as laid out in [Zeig] offers well-characterized levels at which requirements for system behavior can be stated. The process is essentially iterative and leads to increasingly rigorous formulation resulting from the formalization in subsequent phases.
- **Model Structures at higher levels of System Specification:** The formalized behavior requirements are then transformed to the chosen model implementations e.g. DEVS based transformation in C++, Java, C# and others.
- **Simulation Execution:** The model base which may be stored in Model Repository is fed to the simulation engine. It is important to state the fact that separating the Model from the underlying Simulator is necessary to allow independent development of each. Many legacy systems have both the Model and the Simulator tightly coupled to each other which restrict their evolution. DEVS categorically separates the Model from the Simulator for the same simple reason.
- **Real-time Execution:** The simulation can be made executable in real-time mode and in conjunction with Model-Continuity principles, the model itself becomes the deployed code
- **Test Models/Federations:** Branching in the lower-path of the Bifurcated process, the formalized models give way to test models which can be developed at the atomic level or at the coupled level where they become federations. It also leads to the development of experiments and test cases required to test the system specifications. DEVS categorically aids the development of Experimental Frames at this step of development of test-suite.
- **Verification and Validation:** The simulation provides the basis for correct implementation of the system specifications over a wide range of execution platforms and the test suite provides basis for testing such implementations in a suitable test infrastructure. Both of these phases of systems engineering come together in the Verification and Validation (V&V) phase.

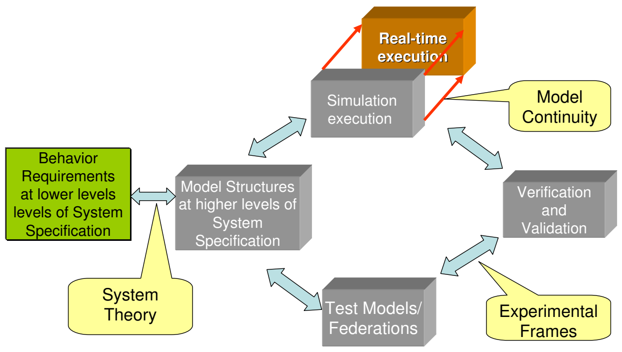

Real-time execution
Simulation execution
Model Continuity
Behavior Requirements at lower levels levels of System Specification
Model Structures at higher levels of System Specification
Verification and Validation
System Theory
Test Models/ Federations
Experimental Frames

Figure 3: Bifurcated Model-Continuity based System Life-cycle Process

3.1 Automated DEVS Model Generation and DEVSML

This section describes various formats in which the system behavior requirements could be expressed for today's systems and the methodologies leading to generation of DEVS models in an automated manner. The requirements are the most important part of any system development and they are seldom specified in a format that is helpful to the developer at large. Consequently, it is refined throughout the system development lifecycle until the developer as well as the stake-holder settles on a common ground. Testing in such iterative developmental cycle bears the burden of 'meeting' the system specifications. To automate both the model generation and test case generation is a current need of the system design process. Consequently, taking first things first, this section enumerates various formats and implementations in which the requirements could be specified. They are as follows:

1. **State-based system specifications:** In this implementation, the system is specified using state-machines with UML [OMG] tools such as Rational Rose, Microsoft Visio or Enterprise Architect . Sometimes the DEVS formalized state machine is also available.
2. **Rule-based system specifications using restricted natural language processing (NLP):** Natural language such as English can be very ambiguous. To make it more specific, either every aspect must be taken into account for every situation, which results in a voluminous record, or the language itself must be restricted with chosen keywords. A restricted NLP is provided and model generation is described [MIT07g]
3. **BPMN/BPEL based system specifications:** Business Process Modeling Notation (BPMN) [bpm] or Business Process Execution Language (BPEL) provide a very high level view of any business process involving sub-processes. This kind of requirement specification is largely graphical in nature and the information is stored in *.wsdl* and *.bpel* files for BPEL but in proprietary format for BPMN.
4. **DoDAF-based requirement specifications:** Department of Defense Architecture Framework (DoDAF) [Dod04c] is the mandated framework for any future government system architecture specification and it suffers from various deficiencies largely attributed to the fact that M&S is not mandated in it.

Having analyzed the information set available in these formats, DEVS information is extracted from these sets. To specify a DEVS system, we have the following basic MUST requirements:

1. Entities as Objects and their hierarchical organization
2. Finite State Machines (FSMs) of atomic models
3. Timeouts for each of the phases (States) in atomic models
4. Entity interfaces as input and output ports
5. External incoming messages at Entity's interface at specified duration in specific State
6. External outgoing messages at Entity's interface at specified duration in specific State
7. Coupling information derived from hierarchical organization and interface specifications
8. Experimental Frame specifications

Various model specification formalisms are supported and mapped into DEVSML models including UML state charts [JLR07], a table driven state-based approach[MIT07h], Business Process Modeling Notation (BPMN) [BPM,BPEL] or DoDAF-based[MIT06a]. A translated DEVSML model is fed to the DEVSML client that coordinates with the DEVSML server farm. Once the client has DEVSJAVA models, a DEVSML server can be used to integrate the client's model with models that are available at other sites to get an enhanced integrated DEVSML file that can produce a coupled DEVSML model. The DEVS/SOA enabled server can use this integrated DEVSML file to deploy the component models to assigned DEVS web-server simulated engines. The integration of DEVSML and DEVS/SOA is performed with the layout as shown below in Figure 4. The result is a distributed simulation, or alternatively, a real-time distributed execution of the coupled model.

3.2 DEVSML Collaborative Development

DEVSML is a way of representing DEVS models in XML language. This DEVSML is built on JAVAML [BAD05], which is XML implementation of JAVA. The current development effort of DEVSML takes its power from the underlying JAVAML that is needed to specify the 'behavior' logic of atomic and coupled models. The DEVSML models are transformable back'n forth to java and to DEVSML. It is an attempt to provide interoperability between various models and create dynamic scenarios. The key concept as a layered architecture is shown in the Figure 5.

At the top is the application layer that contains model in DEVS/JAVA or DEVSML. The second layer is the DEVSML layer itself that provides seamless integration, composition and dynamic scenario construction resulting in portable models in DEVSML that are complete in every respect. These DEVSML models can be ported to any remote location using the net-centric infrastructure and be executed at any remote location. Another major advantage of such capability is total simulator 'transparency'. The simulation engine is totally transparent to model execution over the net-centric infrastructure.

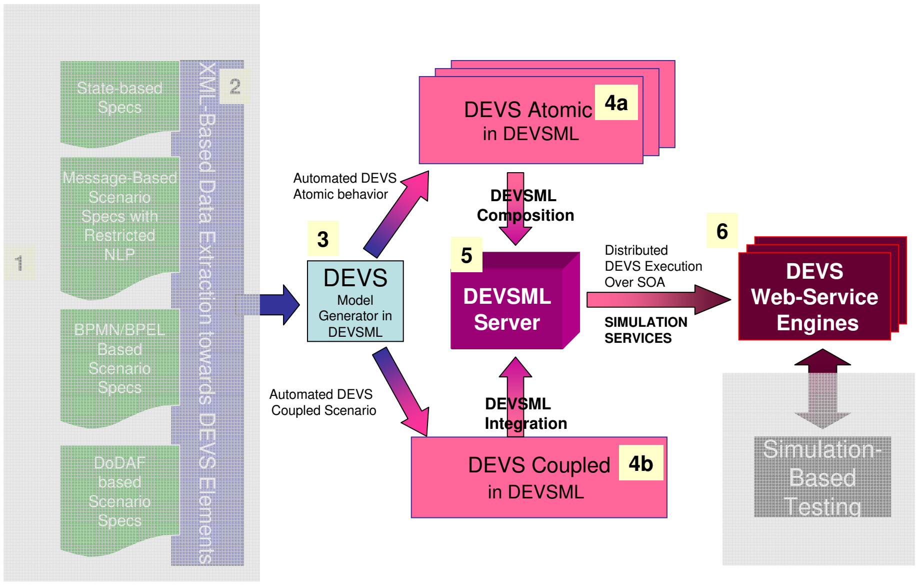

1
State-based Specs
Message-Based Scenario Specs with Restricted NLP
BPMN/BPEL Based Scenario Specs
DoDAF based Scenario Specs
2
XML-Based Data Extraction towards DEVS Elements
3
DEVS Model Generator in DEVSML
Automated DEVS Atomic behavior
Automated DEVS Coupled Scenario
DEVS Atomic in DEVSML 4a
DEVSML Composition
5
DEVSML Server
DEVSML Integration
DEVS Coupled in DEVSML 4b
Distributed DEVS Execution Over SOA
SIMULATION SERVICES
6
DEVS Web-Service Engines
Simulation-Based Testing

Figure 4: Net-centric collaboration and execution using DEVSML and DEVS/SOA

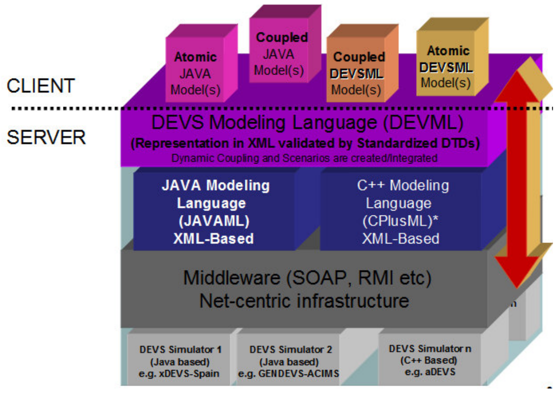

Atomic JAVA Model(s)
Coupled JAVA Model(s)
Coupled DEVSML Model(s)
Atomic DEVSML Model(s)
CLIENT
SERVER
DEVS Modeling Language (DEVML)
(Representation in XML validated by Standardized DTDs)
Dynamic Coupling and Scenarios are created/integrated
JAVA Modeling Language (JAVAML) XML-Based
C++ Modeling Language (CPlusML)* XML-Based
Middleware (SOAP, RMI etc)
Net-centric infrastructure
DEVS Simulator 1 (Java based) e.g. xDEVS-Spain
DEVS Simulator 2 (Java based) e.g. GENDEVS-ACIMS
DEVS Simulator n (C++ Based) e.g. aDEVS

Figure 5: DEVS Transparency and Net-centric model interoperability using DEVSML. Client and Server categorization is done for DEVS/SOA implementation

The DEVSML model description files in XML contains meta-data information about its compliance with various simulation 'builds' or versions to provide true interoperability between various simulator engine implementations. This has been achieved for at least two independent simulation engines as they have an underlying DEVS protocol to adhere to. This has been made possible with the implementation of a single atomic DTD and a single coupled DTD that validates the DEVSML descriptions generated from these two implementations. Such run-time interoperability provides great advantage when models from different repositories are used to compose bigger coupled models using DEVSML seamless integration capabilities. More details about the implementation can be seen at [MIT07e]

3.3 Automated Test-Model Generation from DEVS models

Assuming that the DEVS model is easily specified using State-based approach as described in [JLR07, MIT07h], the automated test-model generation is constructed at Level 1 of Input/Output behavior [Zei00] taking DEVS component as a black-box. The test-model is called the Observer model and its state-machine is defined by the testee component's state-machine. The component model being tested is called Testee and the component model doing the testing is called Tester. The autogeneration mechanism is described in detail in [MIT07g] and is outside of scope of this paper.

3.4 DEVS/SOA: Net-centric Execution using Simulation Service

The fundamental concept of web services is to integrate software application as services. Web services allow the applications to communicate with other applications using open standards. We are offering DEVS-based simulators as a web service, and they must have these standard technologies: communication protocol (Simple Object Access Protocol, SOAP), service description (Web Service Description Language, WSDL), and service discovery (Universal Description Discovery and Integration, UDDI).

The complete setup requires one or more servers that are capable of running DEVS Simulation Service, as shown in Figure 6. The capability to run the simulation service is provided by the server side design of DEVS

Simulation protocol supported by the latest DEVSJAVA Version 3.1[ACI06]. The Simulation Service framework is two layered framework. The top-layer is the user coordination layer that oversees the lower layer. The lower layer is the true simulation service layer that executes the DEVS simulation protocol as a Service.

From multifarious modes of DEVS model generation, the next step is the simulation of these models. The DEVS/SOA client takes the DEVS models package and through the dedicated servers hosting simulation services, it performs the following operations:

1. Upload the models to specific IP locations i.e. partitioning
2. Run-time compile at respective sites
3. Simulate the coupled-model
4. Receive the simulation output at client's end

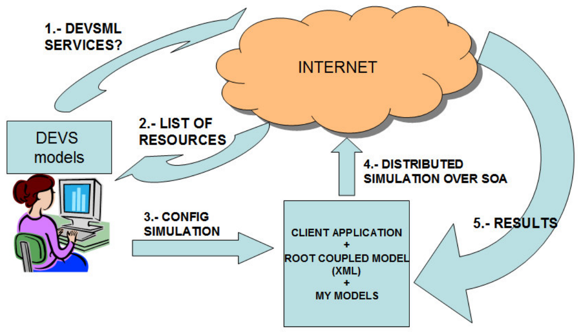
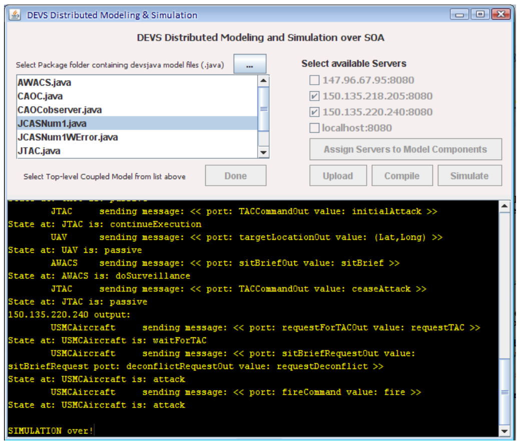

1.- DEVSML SERVICES?
INTERNET
2.- LIST OF RESOURCES
DEVS models
4.- DISTRIBUTED SIMULATION OVER SOA
3.- CONFIG SIMULATION
CLIENT APPLICATION + ROOT COUPLED MODEL (XML) + MY MODELS
5.- RESULTS

Figure 6: Execution of DEVS SOA-Based M&S

The main server selected (corresponding to the top-level coupled model) creates a coordinator that creates simulators in the server where the coordinator resides and/or over the other remote servers selected. The DEVS/SOA web service client as shown in Figure 7 operates in the following sequential manner:

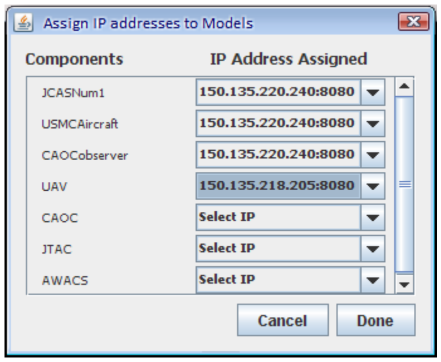
1. The user selects the DEVS package folder at his machine
2. The top-level coupled model is selected as shown in Figure 7
3. Various available servers are selected. Any number of available servers can be selected. Figure 8 shows how Servers are allocated on per-model basis. The user can specifically assign specific IP to specific models at the top-level coupled domain. The localhost (Figure 7) is chosen using debugging sessions.
4. The user then uploads the model by clicking the Upload button. The models are partitioned in a round-robin mechanism and distributed among various chosen servers
5. The user then compiles the models by clicking the Compile button at server's end
6. Finally, Simulate button is pressed to execute the simulation using Simulation service hosted by these services.
7. Once the simulation is over, the console output window displays the aggregated simulation logs from various servers at the client's end.


DEVS Distributed Modeling & Simulation
DEVS Distributed Modeling and Simulation over SOA
Select Package folder containing devsjava model files (.java)
...
AWACS.java
CAOC.java
CAOCobserver.java
JCASNum1.java
JCASNum1WError.java
JTAC.java
Select Top-level Coupled Model from list above
Done
Select available Servers
147.96.67.95:8080
150.135.218.205:8080
150.135.220.240:8080
localhost:8080
Assign Servers to Model Components
Upload
Compile
Simulate

```
        JTAC     sending message: << port: TACCommandOut value: initialAttack >>
State at: JTAC is: continueExecution
        UAV      sending message: << port: targetLocationOut value: (Lat,Long) >>
State at: UAV is: passive
        AWACS    sending message: << port: sitBriefOut value: sitBrief >>
State at: AWACS is: doSurveillance
        JTAC     sending message: << port: TACCommandOut value: ceaseAttack >>
State at: JTAC is: passive
150.135.220.240 output:
        USMCAircraft     sending message: << port: requestForTACOut value: requestTAC >>
State at: USMCAircraft is: waitForTAC
        USMCAircraft     sending message: << port: sitBriefRequestOut value:
sitBriefRequest port: deconflictRequestOut value: requestDeconflict >>
State at: USMCAircraft is: attack
        USMCAircraft     sending message: << port: fireCommand value: fire >>
State at: USMCAircraft is: attack

SIMULATION over!
```

Figure 7: GUI snapshot of DEVS/SOA client hosting distributed simulation



Components	IP Address Assigned
JCASNum1	150.135.220.240:8080
USMCAircraft	150.135.220.240:8080
CAOCobserver	150.135.220.240:8080
UAV	150.135.218.205:8080
CAOC	Select IP
JTAC	Select IP
AWACS	Select IP

Figure 8: Server Assignment to Models

In terms of net-ready capability testing, what is required is the communication of live web services with those of test-models designed specifically for them. The approach has the following steps:

1. Specify the scenario
2. Develop the DEVS model
3. Develop the test-model from DEVS models
4. Run the model and test-model over SOA
5. Execute as a real-time simulation
6. Replace the model with actual web-service as intended in scenario.
7. Execute the test-models with real-world web services
8. Compare the results of steps 5 and 7.

4. The Complete DUNIP Process

DUNIP can be summarized as the sequence of the following steps:

1. Develop the requirement specifications in one of the chosen formats such as BPMN, DoDAF, Natural Language Processing (NLP) based, UML based or simply DEVS-based for those who understand the DEVS formalism
2. Using the DEVS-based automated model generation process, generate the DEVS atomic and coupled models from the requirement specifications using XML or XFDDEVS [MIT07h]
3. Validate the generated models using DEVS W3C atomic and coupled schemas to make them net-ready capable for collaborative development, if needed. This step is optional but must be executed if distributed model development is needed. The validated models which are Platform Independent Models (PIMs) in XML can participate in collaborative development using DEVSML.
4. From step 2, either the coupled model can be simulated using DEVS/SOA or a test-suite can be generated based on the DEVS models.
5. The simulation can be executed on an isolated machine or in distributed manner using SOA middleware if the focus is net-centric execution. The simulation can be executed in real-time as well as in logical time.
6. The test-suite generated from DEVS models can be executed in the same manner as laid out in Step 5.
7. The results from Step 5 and Step 6 can be compared for V&V process.

The basic Bifurcated Model Continuity-based Life-cycle process for systems engineering in Figure 3 in light of the developments in DEVS area is summarized in Figure 9 below. The grey boxes show the original process and the colored boxes show the extensions that were developed to make it a DEVS compliant process. A sample demo movie is available at [Dun07].

Many case studies came about as DUNIP was defined and developed. Many of the projects are currently active at Joint Interoperability Test Command (JITC) and others are at concept validation stage towards a deliverable end. Each of the projects either uses the complete DUNIP process or a subset of it. As we shall see on a case by case basis, DEVS emerge as a powerful M&S framework contributing to the complete systems software engineering process. With the proposed DEVS Based Bifurcated Model-continuity Life-cycle process, systems theory with its DEVS implementation can support the next generation net-centric application development and testing.

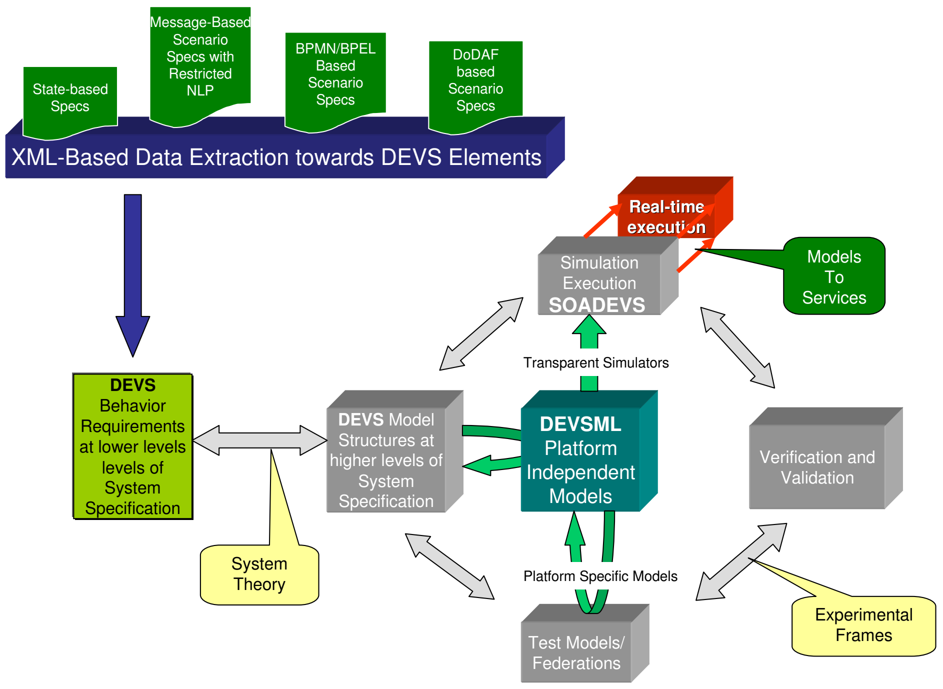

State-based Specs
Message-Based Scenario Specs with Restricted NLP
BPMN/BPEL Based Scenario Specs
DoDAF based Scenario Specs
XML-Based Data Extraction towards DEVS Elements
Real-time execution
Simulation Execution SOADEVS
Models To Services
Transparent Simulators
DEVS Behavior Requirements at lower levels levels of System Specification
DEVS Model Structures at higher levels of System Specification
DEVSML Platform Independent Models
Verification and Validation
System Theory
Platform Specific Models
Test Models/ Federations
Experimental Frames

Figure 9: The Complete DEVS Unified Process

A recent Doctoral Thesis [MIT07g] provides a number of examples that illustrate how the DUNIP can be applied to import real-world simulation-based design applications. Here we review briefly the following case studies that were developed in more detail in [MIT07g]:

- Joint Close Air Support (JCAS) model
- DoDAF-based Activity scenario
- Link-16 Automated Test Case Generator (ATC-Gen project at JITC)
- Generic Network for Systems Capable of Planned Expansion (GENETSCOPE project at JITC)

Each of the projects has been developed independently and ATC-Gen and GENETSCOPE are team projects. All of the projects stand-alone and each applies DUNIP (Table 1 in full or in-part). Table 1 below provides an overview of the DUNIP elements used in each of the projects. All of the DUNIP elements have been applied at least once in one of the projects.

The JCAS system requirements come in many formats and served as a base example to test many of the DUNIP processes for requirements-to-DEVS transformation [MIT07g]. JCAS requirements were specified using the state-based approach, BPEL-based approach and restricted natural language approach. The JCAS case study describes how each of the three approaches led to executable DEVS models with identical simulation results. Finally, the simplest executable model (that specified by the state-based approach) was executed over a net-centric platform using DEVSML and DEVS/SOA architecture.

The DODAF-based Activity scenario was specified using the UML-based Activity diagrams. It illustrates the process needed to transform various DoDAF documents into DEVS requirement specifications. New Operational View documents OV-8 and OV-9 were proposed [Mit06a] to facilitate the transformation of DoDAF requirements into a form that could be supported by DEVS-based modeling and simulation. The population of these new documents was described as well as how DEVS models could be generated from them.

The ATC-Gen project at JITC is the project dealing with automated Link-16 testing environment and the design of ATC-Gen tool. A detailed discussion and complete example are presented in [MAK06].

The GENETSCOPE project [GEN06] at JITC is another project that employs the complete DEVS software engineering process. Using automated XML data mining, a ten year-old legacy model written in the C language was transformed to an object-oriented DEVS model with enhanced Model View Simulation and Control paradigm [Mit06b]. The design elements of GENETSCOPE tool were discussed and as was its relationship with the overarching DoDAF framework [Mit06b].

Project / DUNIP Elements		**JCAS model**	**DoDAF-based Activity Scenario**	**ATC-Gen Project**	**GenetScope Project**
Requirement Specification Formats		X			X
	State-based Specs	X			
	Message-based Specs with restricted NLP	X			
	BPMN/BPEL based Specs	X			
	DoDAF-Based Scenario Specs		X		X
XML-based Data Extraction		X	X	X	
DEVS Model Structure at lower levels of Specification		X	X	X	
DEVS model structure at higher levels of System specification			X		X
DEVSML Platform Independent Models		X			
Test Model Development		X		X	
Verification and Validation using Experimental Frames			X	X	X
DEVS/SOA net-centric Simulation		X			

Table 1: Overview of DUNIP application in available case-studies

5. Discussion

This section discusses the DUNIP process with current state of the art in model-based engineering processes. Two paradigms have been chosen: MDA and SCR. MDA or Model-Driven Architecture is philosophy as put forward by Object Modeling Group (OMG) that comprises of many standards like UML, XMI, Meta-Object Facility (MOF) and others. SCR is the Software Cost Reduction methodology by Hartmeyer.

5.1 MDA and DUNIP

DUNIP is built on the paradigm of Model-Based Engineering, or Model Driven Architecture (MDA). However, the scope of DUNIP goes beyond the MDA objectives. Potential concerns with the current MDA state of art include:

- MDA approach is underpinned by a variety of technical standards, some of which are yet to be specified (e.g. executable UML)
- Tools developed my many vendors are not interoperable
- MDA approach is considered too-idealistic lacking iterative nature of Software Engineering process
- MDA practice requires skilled practitioners and design requires engineering discipline not commonly available to code developers.

Further, MDA does not have any underlying Systems theory and groups like INCOSE[1] are working with OMG to adapt UML to systems engineering. Testing is included only as an extension of UML, known as executable UML [Mel02], for which there is no current standard. Consequently, there is no testing framework that binds executable UML and simulation-based testing.

Despite these shortcomings, MDA has been adopted by Joint Single Integrated Air Picture (SIAP) Systems Engineering Organization (JSSEO) and various recommendations have come forth to enhance the MDA process. JSSEO is applying MDA approach toward development of aerospace Command and Control (C2) capabilities, for which a single integrated air picture is foundational. The data-driven nature of C2 System of Systems (SoS) means that powerful MDA concepts adapt well to collaborative SoS challenges.

Current DoD enterprise-level approaches for managing SoS interoperability, like the Net Centric Operations and Warfare Reference Model (NCOW/RM), DoD Architecture Framework (DoDAF) and the Joint Technical Architecture (JTA), simply do not have the technical strength to deal with the extremely complex engineering challenges [Jac04]. We proposed enhanced DoDAF [Mit06a] to provide DEVS-based Model engineering. MDA as implemented by industry and adapted by JSSEO, does have the requisite technical power, but requires innovative engineering practices.

Realizing the importance of MDA concepts and the executable profile of UML, the basic objective of which is to simulate the model, JSSEO is indirectly looking at the Modeling & Simulation domain as applicable to SoS engineering. The following table brings out the shortcomings of MDA in its current state and the capabilities provided by DEVS technology and in turn, DUNIP process.

MDA as applied to Integration of Process-Driven SOA Models

In an independent study [Zdu02], Model Driven Software Development (MDSD) was applied to the integration of process-driven SOA models. UML2 was used as the basis towards integration. Their approach is based on the notion of domain-specific languages (DSL) for modeling various types of models. Once DSL has been identified, its meta-model is created that represents this particular modeling domain. Meta-models are defined in terms of meta-meta-model. In UML, this is the meta object facility (MOF). They created a meta-meta-model that would define both the UML2 meta-model and their selected DSL extensions. The whole objective is to find a common ground and a way to express the relationship between a meta-model and the implementation code. This kind of capability where a single meta-meta-model can be used to integrate two different DSLs towards a common model allowing specific constraints of each meta-model is very much needed in SOA domain as multiple tools and standards exist preventing such integration. To integrate two models with different DSLs, the models are first decomposed at the meta-model level, required information extracted and supplemented (on the basis of meta-meta-model), which results in an integrated model. In our DUNIP process, such collaboration comes naturally due to the proposed DEVS atomic and coupled Document Type Definitions (DTDs) that specify any DEVS model in any domain specific language implementations. The underlying DEVS Modeling Language (DEVSML) meta-model that defines these atomic and coupled DTDs is used for validating any DEVS model. The current DEVSML implementation has successfully integrated two DSL implementations (GenDEVS-ACIMS and xDEVS-Spain) on common DEVSML atomic and coupled DTDs. These two DTDs have also been submitted to

[1] International Council on Systems Engineering

DEVS standardization group towards standardization. Their extension towards DEVS atomic and coupled schemas are available online at [DASc,DCSc].

5.2 DUNIP and SCR

Software Cost Reduction (SCR) method allows development of formal requirements using a tabular notation. The SCR toolset includes an editor for building the specifications, a consistency checker for testing the specifications for consistency with formal requirements model, a simulator for symbolically executing the specifications and a verifier for checking that the specifications satisfy selected applications properties [Heit95]. SCR has been used to define requirements for embedded systems as well as software systems. SCR is more exhaustive and complete in terms of model checking and consistency checking. It is at a higher order of resolution where state variables can be a part of the specification definition.

In DUNIP, although it is based on DEVS, the state-variables are not considered in the automated DEVS model generation as described in Chapter 4. Our current work falls in the category of a subset of DEVS specifications, where only message passing between the components is considered. The motivation of this research effort stems from the need of absence of an M&S framework for Net-centric systems collaborating over the GIG. The systems are at a much higher level of abstractions than any embedded system where state-variable bear much importance and criticalities. The current version of DUNIP addresses the need of these abstract systems. Inclusion of state-variables, more like on the lines of SCR will be included in future, to develop more sophisticated models.

Desired M&S Capability	**MDA**	**DUNIP**
Need for executable architectures using M&S	Yes, although not a standard yet	Yes, underlying DEVS theory [ZEI00]
Applicable to GIG SOA	Not reported yet	Yes
Interoperability and cross-platform M&S using GIG/SOA	--	Yes, DEVSML and DEVS/SOA provides cross-platform M&S using Simulation Web Services [MIT07e,f,g]
Automated test generation and deployment in distributed simulation	--	Yes, based on formal Systems theory and test-models autogeneration at various levels of System specifications [Zei05. MIT07g]
Test artifact continuity and traceability through phases of system development	To some extent, model becomes the application itself	Yes
Real time observation and control of test environment	--	Dynamic Model Reconfiguration and run-time simulation control integral to DEVS M&S. Enhanced MVC framework is designed to provide this capability [Mit06b]

Table 2: Comparison of MDA and DUNIP

5.3 Potential Issues and Drawbacks

Building upon and integrating earlier systems theoretic and architectural methodologies, DUNIP inherits many of the advantages that such methodologies afford and attempts to fill in the holes that they still leave. However, as with the methodologies it draws upon, and is inspired by, there are potential issues and drawbacks that may be expected to emerge in its application. The complexity and quality assurance issues associated with the proposed methodology need to be mitigated with the development of appropriate tools and interfaces to simplify working with the methodology. The need for such complexity-reduction tools underlies the extended discussions of tools and interfaces that have been provided here. Further quality assurance demands provision of approaches to self-checking DUNIP and its supporting infrastructures.

Finally, there remain many issues to resolve in the manner in which the DUNIP methodology relates to the defense-mandated DODAF. The latter is a representational mechanism, not a methodology, and does not discuss how an integrated architecture should be constructed and evolved from existing systems. DUNIP offers an approach based on systems theory and supported by DEVS-based modeling and simulation to tackle integration and interoperability issues, but the integration with DODAF remains for future consideration.

6. Conclusion

The proposed methodology can be used for integrating heterogeneous constituents of an SoS and assessing their real time interactions and interoperability. The proposed methodology encompasses the advantages of several interrelated concepts such as the systems theory; DEVSML and DEVS/SOA; M&S framework; and the model continuity concepts. Especially, since it separates models from their underlying simulators; enables real time execution and testing at multiple levels and over a wide rage of execution platforms; uses open standards; supports collaborative development; and has the potential to provide additional SoS architectural views. Although the DUNIP was applied in part to many projects, presently, there is no live case study that implements all the aspects of DUNIP elements. Recall that DUNIP was researched and developed in the context of many active projects at JITC and ACIMS.

The SoS can be specified in many available frameworks such as DoDAF, system theory, UML, or by using an integrative systems engineering-based framework such as DEVS. In this paper, we have discussed the advantages of employing an M&S-integrated framework such as DEVS Unified Process (DUNIP) and its supporting DEVS/SOA infrastructure. As components comprising SoS are designed and analyzed, their integration and communication is the most critical part that must be addressed by the employed SoS M&S framework. The DEVS Unified Process (DUNIP), in analogy to the Rational Unified Process based on UML, offers a process for integrated development and testing of systems that rests on the SOA infrastructure. The DUNIP perspective led us to formulate a methodology for testing any proposed SOA-based integration infrastructure, such as DISA's Net-Centric Enterprise Services.

7. References

[ACI06] ACIMS software site: http://www.acims.arizona.edu/SOFTWARE/software.shtml Last accessed April 2008

[ATK04] K. Atkinson, "Modeling and Simulation Foundation for Capabilities Based Planning", Simulation Interoperability Workshop Spring 2004

[BAD05] Badros, G. JavaML: a Markup Language for Java Source Code, Proceedings of the 9th International World Wide Web Conference on Computer Networks: the international journal of computer and telecommunication networking, pages 159-177

[BPMN] Business Process Modeling Notation (BPMN) www.bpmn.org

[BPEL] Business Process Execution Language (BPEL) http://en.wikipedia.org/wiki/BPEL

[CAR05] Carstairs, D.J., "Wanted: A New Test Approach for Military Net-Centric Operations", Guest Editorial, ITEA Journal, Volume 26, Number 3, October 2005

[Cho01] Cho, Y., B.P. Zeigler, H.S. Sarjoughian, Design and Implementation of Distributed Real-Time DEVS/CORBA, IEEE Sys. Man. Cyber. Conf., Tucson, Oct. 2001.

[CJC04] Chairman, JCS Instruction 3170.01D "Joint Capabilities Integration and Development System," 12 March 2004.

[CJC06] Chairman, JCS Instruction 6212.01D "Interoperability and Supportability of Information Technology and National Security Systems," March 8, 2006, 271

[DASc] DEVS Atomic Schema: http://www.acims.arizona.edu/EDUCATION/ECE676Spring08/NewXMLSchema.xsd

[DCSc] DEVS Coupled Schema: http://www.acims.arizona.edu/EDUCATION/ECE676Spring08/CoupledDevs.xsd

[DOD03a] DoDAF Working Group , DoD Architecture Framework Ver. 1.0 Vol. III: Deskbook, DoD, Aug. 2003.

[DOD03b] DOD Instruction 5000.2 "Operation of the Defense Acquisition System," 12 May 2003.

[DOD04c] DoD Architecture Framework Working Group 2004, DOD Architecture Framework Ver. 1.0 Volume 1 Definitions and Guidelines, 9 February 2004, Washington, D.C.

[DUN07] DUNIP: A Prototype Demonstration http://www.acims.arizona.edu/dunip/dunip.avi

[GEN06] GENETSCOPE(Beta Version) Software User's Manual, available from ACIMS center, University of Arizona.

[Hiet95] Heitmeyer, C., Bull, A., Gasarch, C., Labaw, B., SCR: A Toolset for Specifying and Analyzing Requirements. Proceedings of the Tenth Annual Conference on Computer Assurance (COMPASS '95), Gaithersburg, MD, June 25-29, 1995, pp. 109-122

[Jac04] Jacobs, R., Model-Driven Development of Command and Control Capabilities For Joint and Coalition Warfare, Command and Control Research and Technology Symposium, 2004

[JLR07] Martin, JLR, Mittal, S., Zeigler, B.P., Manuel, J., "From UML Statecharts to DEVS State Machines using XML", IEEE/ACM conference on Multi-paradigm Modeling and Simulation, Nashville, September 2007

[MAK06] Mak, E., Automated Testing using XML and DEVS, Thesis, University of Arizona, http://www.acims.arizona.edu/PUBLICATIONS/PDF/Thesis_EMak.pdf

[MAK07] E Mak, S Mittal, MH Hwang, "Automating Link-16 Testing using DEVS and XML", Journal of Defense Modeling and Simulation, to appear

[MIT06a] Mittal, S. "Extending DoDAF to Allow DEVS-Based Modeling and Simulation", Special issue on DoDAF, Journal of Defense Modeling and Simulation JDMS, Vol 3, No. 2.

[Mit06b] Mittal.S., Mak, E. Nutaro, J.J., "DEVS-Based Dynamic Modeling & Simulation Reconfiguration using Enhanced DoDAF Design Process", special issue on DoDAF, Journal of Defense Modeling and Simulation, Dec 2006

[Mit07c] Mittal, S., Risco-Martin, J.L., Zeigler, B.P. "DEVS/SOA: A Cross-Platform framework for Net-Centric Modeling and simulation in DEVS Unified Process", submitted to SIMULATION, under review.

[Mit07e] Mittal, S., Risco-Martin, J.L., Zeigler, B.P. "DEVSML: Automating DEVS Simulation over SOA using Transparent Simulators", DEVS Syposium, 2007

[Mit07f] Mittal, S., Risco-Martin, J.L., Zeigler, B.P. "DEVS-Based Web Services for Net-centric T&E", Summer Computer Simulation Conference, 2007

[MIT07g] Mittal, S, DEVS Unified Process for Integrated Development and Testing of Service Oriented Architectures, Ph. D. Dissertation, University of Arizona

[MIT07h] Mittal, S., , M.H., Zeigler, B.P., Hwang "XFD-DEVS: An Implementation of W3C Schema for Finite Deterministic DEVS", in progress, Demo available at: http://www.saurabh-mittal.com/fddevs

[OMG] Object Modeling Group (OMG) www.omg.org

[SAR00] Sarjoughian, H.S., B.P. Zeigler, "DEVS and HLA: Complimentary Paradigms for M&S?" Transactions of the SCS, (17), 4, pp. 187-197, 2000

[SAR01] H. Sarjoughian, B. Zeigler, and S. Hall, *A Layered Modeling and Simulation Architecture for Agent-Based System Development*, Proceedings of the IEEE 89 (2); 201-213, 2001

[UML] Unified Modeling Language (UML), http://www.omg.org/technology/documents/formal/uml.htm

[Wai01] Wainer, G., Giambiasi, N., Timed Cell-DEVS: modeling and simulation of cell-spaces". Invited paper for the book Discrete Event Modeling & Simulation: Enabling Future Technologies, Springer-Verlag 2001

[Weg02] Wegmann, A., "Strengthening MDA by Drawing from the Living Systems Theory", Workshop in Software Model Engineering, 2002

[Zha05] Zhang, M., Zeigler, B.P., Hammonds, P., DEVS/RMI-An Auto-Adaptive and Reconfigurable Distributed Simulation Environment for Engineering Studies, ITEA Journal, July 2005

[Zdu02] Zdun, U., Dustdar, S. (2007). Model-Driven Integration of Process-Driven SOA Models. *International Journal of Business Process Integration and Management*, Inderscience,

[ZEI00] Zeigler, B. P., T. G. Kim, and H. Praehofer. (2000). Theory of Modeling and Simulation. New York, NY, Academic Press.

[ZEI05] Zeigler, B.P., Fulton, D., Hammonds P., and Nutaro, J. Framework for M&S–Based System Development and Testing In a Net-Centric Environment, ITEA Journal of Test and Evaluation, Vol. 26, No. 3, 21-34, 20

Biography

SAURABH MITTAL is an Assistant Research Professor at the ECE Department, University of Arizona. He received both MS and PhD in ECE from the University of Arizona in 2004 and 2007 respectively. His research interests include modeling and simulation, net-centric systems engineering, DoDAF-based executable architectures, interoperability and data engineering. He is a recipient of Joint Interoperability Test Command's highest civilian contractor 'Golden Eagle' award for the project GENETSCOPE and NTSA award for Best Cross-platform development in M&S area for the project ATC-Gen. He is currently working on projects at JITC and NGIT.

BERNARD P. ZEIGLER is Professor of Electrical and Computer Engineering at the University of Arizona, Tucson and Director of the Arizona Center for Integrative Modeling and Simulation. He is internationally known for his 1976 foundational text Theory of Modeling and Simulation, recently revised for a second edition (Academic Press, 2000), He has published numerous books and research publications on the Discrete Event System Specification (DEVS) formalism. In 1995, he was named Fellow of the IEEE in recognition of his contributions to the theory of discrete event simulation. In 2000 he received the McLeod Founders Award by the Society for Computer Simulation, its highest recognition, for his contributions to discrete event simulation. He was appointed Fellow of the Society for Modeling and Simulation, International (SCS), 2006.