DEVS Standard for Modeling and Simulation in Web-Centric Environments

Bernard P. Zeigler
Arizona Center for Integrative Modeling and Simulation
University of Arizona

Presented at

Mosim08: Modélisation, Optimisation et Simulation des Systèmes

March 31-April 2 2008
Paris, France
Modeling and Simulation (M&S) Framework

Experimental Frame

Source System

Modeling Relation

Model

Simulator

Simulation Relation
A discrete event system specification (DEVS) is a structure
\[ M = \langle X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \delta_{\text{con}}, \lambda, \tau_a \rangle \]
where
- \( X \) is the set of input values,
- \( S \) is a set of states,
- \( Y \) is the set of output values,
- \( \delta_{\text{int}} : S \rightarrow S \) is the internal transition function,
- \( \delta_{\text{ext}} : Q \times X \rightarrow S \) is the external transition function,
- \( \delta_{\text{con}} : Q \times X \rightarrow S \) is the confluent transition function,
- \( \tau_a : S \rightarrow R^*_0,\infty \)

Where
- \( Q = \{(s, e) | s \in S, 0 \leq e \leq \tau_a(s) \} \) is the total state set,
- \( e \) is the time elapsed since last transition,
- \( \lambda : S \rightarrow Y \) is the output function and
- \( R^*_0,\infty \) is the set of positive reals with 0 and \( \infty \)
DEVS Hierarchical Modular Models

System Entity Structure

Hierarchical Composition, Coupling and Variants are Represented in System Entity Structure
Advantages of DEVS

Theory
- Closure under coupling, universality, uniqueness, relation to other formalisms
- Hierarchical Model Construction supports complex systems
- Supporting the correctness of the algorithms and validation of the executing models

Application
- Models, Simulators and Experimental Frames are distinct entities with their own software representations.
- Precise and well-defined mathematical representation
- Models/Experiments are developed systematically for interoperability
- Repositories of models/experiments are created and maintained systematically (and existing components can be easily reused for constructing new models)
- Discrete-event basis improves performance (e.g. no need for have a global clock to control timing)
Some Types of Models Represented in DEVS

Atomic Models

Ordinary Differential Equation Models

Processing/Queuing/Coordinating

Spiking Neuron Models

Petri Net Models

Discrete Time/StateChart Models

Stochastic Models

Quantized Integrator Models

Fuzzy Logic Models

Coupled Models

Networks, Collaborations

Processing Networks

Spiking Neuron Networks

Physical Space

n-Dim Cell Space

Cellular Automata

Multi Agent Systems

Self Organized Criticality Models

can be components in a coupled model
DEVS Research Groups/Environments

- Carleton’s CD++,
- ADEVS (ORNL *),
- DEVS/C#,
- DEVS/HLA,
- DEVSJAVA (ACIMS - University of Arizona *),
- GALATEA (USB – Venezuela),
- LSIS (Aix-Marseille III – France *),
- JDEVS (Université de Corse - France), PyDEVS (McGill),
- PowerDEVS (University of Rosario, Argentina),
- SimBeams (University of Linz – Austria),
- VLE (Université du Litoral -France),
- SmallDEVS (Brno University of Technology, Czech Republic),
- James (University of Rostock, Germany)
- Portugal, Spain, and Russia;

- **Workshop on Net-Centric Modeling & Simulation**
  March 6–7 2008 - Marseille, France
- [http://osa.inria.fr/wiki/NCMS/NCMS](http://osa.inria.fr/wiki/NCMS/NCMS)
DEVS Adopters

- Joint Interoperability Test Command, USA
- Air Force, Navy / USA, South Korea
- Lockheed Martin Missile Systems
- Usinor – Sachem Expert Control
- Swedish Materials Command
- ...
Global Information Grid / Service Oriented Architecture

Net-Enabled Command & Control

NCES: Secure, agile, robust, dependable, interoperable data-sharing environment for DOD where warfighter, business, and intelligence users share knowledge on a global network. This, in turn, facilitates information superiority, accelerates decision-making, effective operations and net-centric transformation.
Service Oriented Architecture Basics

Search find_xxx

Post save_xxx

Content/Service Catalogs/Registries

WSDL

(Content/Service Consumer)

(Content/Service Consumer)

(Content/Service Provider)

Client Access (& Use)

Service

Simple Object Application Protocol

XML Schema

XML Payload

 SOAP

Client Access (& Use)
Requirements for Testing and Data Collection

- Testing for Organization and Ontology quality
- Verification/Validation relative to service
- Assessment of content for pragmatic, semantic, syntactic correctness
- Measurement of timeliness of information exchange

Content discovery accuracy and effectiveness

Search find_xxx

Post save_xxx

(Bind)

Simple Object Application Protocol

SOAP
Net-Centric Test Agent Capability (NTAC)

Content/Service Catalogs/Registries

- Content/Service Consumer
- Content/Service Provider
- Client Access (& Use)
- Service

- XML Schema
- XML Payload

- SOAP

WSDL

- Test Director
- Data Analyst

Embedded "Test Agent"

- Real time test data, status
- Post test data / metrics for analysis

Agent-to-Agent communication/coordination
### Levels of System of System Interoperability

<table>
<thead>
<tr>
<th>Linguistic Level</th>
<th>Interoperability Demonstrated if:</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pragmatic</strong> – How information in message is used</td>
<td>The receiver reacts to the message in a manner that the sender intends</td>
<td>A commander’s order is obeyed by the troops in the field as the commander intended. (This assumes semantic interoperability.)</td>
</tr>
<tr>
<td><strong>Semantic</strong> – Shared understanding of meaning of messages</td>
<td>The receiver assigns the same meaning as the sender did to the message.</td>
<td>An order from a commander to multinational participants in a coalition operation is understood in the same manner despite translation into different languages.</td>
</tr>
<tr>
<td><strong>Syntactic</strong> – Common rules governing composition and transmitting of messages</td>
<td>The consumer is able to receive and parse the sender’s message</td>
<td>A common network protocol (e.g., IPv4) ensures that all nodes on the network can send and receive data bit arrays while adhering to a prescribed format.</td>
</tr>
</tbody>
</table>
Mapping M&S Layers to Linguistic Levels

- **Syntactic Level**
  - Execution Layer
    - Abstract Simulators, Real time Execution, Animation, Visualization

- **Semantic Level**
  - Modeling Layer
    - Ontologies, Formalisms, Model Dynamic Structure, Life Cycle Continuity, Model Abstraction

- **Pragmatic Level**
  - Network Layer
    - Distributed Grids, Service Oriented Architectures
  - Design and Test Development Layer
    - SES, DoDAF, Integrated System Development and Testing
  - Experimental Frame Layer
    - Observers and Agents for Net-Centric Key Performance Parameters
  - Collaboration Layer
    - Semantic Web, Composition, Orchestration
High level agents collaborate to control and observe mission thread executions.

Higher level agents inform lower level agents of the objectives for health monitoring.

Middle layer alert higher layer agents of network conditions that invalidate test results.

Middle layer agents activate probes at lower layer.

Network probes return statistics and alarms to higher layer agents.
Concept of DEVS Standard

- Single processor
- Distributed Simulator
- Real-Time Simulator
- Virtual-Time Simulator

- DEVS Simulation Protocol
- DEVS Simulator Interface
- Non DEVS

- C++
- Java
- DEVSML

- Other Representation
DEVS Simulation Protocol

Coordinator

simulators.tellAll("initialize")
simulators.AskAll("nextTN")
simulators.tellAll("computeInputOutput")
simulators.tellAll("sendMessages")
simulators.tellAll("ApplyDeltFunc")

DEVS Simulator

DEVS Model

DEVS Simulator

DEVS Model

Non-DEVS Simulator

putContentOnSimulator

1

2

?
interface coreSimulatorInterface{
    void setSimulators
        (Collection<CoreSimulatorInterface>);
    void initialize();
    Double nextTN();
    void computeInputOutput(Double t);
    void applyDeltFunc(Double t);
    void putContentOnSimulator
        (CoreSimulatorInterface sim, ContentInterface c);
    void sendMessages();
}
Finite Deterministic DEVS : FD-DEVS

FDDEVS = <incomingMessageSet, outgoingMessageSet, StateSet, TimeAdvanceTable, InternalTransitionTable, ExternalTransitionTable, OutputTable>
where
incomingMessageSet, outgoingMessageSet, StateSet are finite sets
TimeAdvanceTable: StateSet → \( R_{0,\infty}^+ \) (the positive reals with zero and infinity)
InternalTransitionTable: StateSet → StateSet
ExternalTransitionTable: StateSet × incomingMessageSet → StateSet,
OutputTable: StateSet → the set of subsets of outgoingMsgSet

Natural Language For FDDEVS
• to start hold in PHASE for time SIGMA
• when in PHASE and receive MSG go to PHASE’
• hold in PHASE for time SIGMA
• after PHASE then output MSG
• from PHASE go to PHASE’

Semantics defined by mapping into DEVS
FD-DEVS

- The “right” abstraction of DEVS – retains important timing properties
- Amenable to analysis
- Supports automation
- Maps to DEVSJAVA
- Supplies a skelton that can be extended to full DEVS
- Simple XML expression

Example of Natural Language Spec:
1. to start passivate in waitForStart
2. when in waitForStart and receive StartAck go to sendFirst
3. hold in sendFirst for time 0.5 then output ReqForAck and go to sendSecond
4. hold in ackReceived for time Infinity
5. last Chance

ReqForAck
Automated Analysis: Based on State Input Pairs Enables Automated Test Model Generation

Defined State-Input pairs:
- listen , orderOffer
- awaitingAdv , ProdFeasibilityFalse
- awaitingAdv , ProdFeasibilityTrue
- waitForConfirm , Confirm

Defined State-Input pairs:
- listen , orderOffer
- awaitingAdv , ProdFeasibilityFalse
- awaitingAdv , ProdFeasibilityTrue
- waitForConfirm , Confirm

Terminal State-Input Pairs:
- awaitingAdv , ProdFeasibilityFalse
- waitForConfirm , Confirm

State-Input pair paths:
1 = listen , OrderOffer
2 = listen , OrderOffer, awaitingAdv, ProdFeasibilityFalse
3 = listen , OrderOffer, awaitingAdv, ProdFeasibilityTrue
4 = listen , OrderOffer, awaitingAdv, ProdFeasibilityTrue, waitForConfirm, Confirm

State-Input Pairs mapped To Terminal State-Input pairs:
- listen , orderOffer -> awaitingAdv , ProdFeasibilityFalse
- awaitingAdv , ProdFeasibilityTrue -> waitForConfirm , Confirm

Dominant State-Input pair:
- listen , orderOffer

OrderOffer

listen

Awaiting Adv

waitForConfirm

Confirm

ProdFeasibilityTrue

ProdFeasibilityFalse
Generated Test Models in DEVSJAVA SimView
System Entity Structure (SES) : SESBuilder
System Entity Structure/Model Base Repository: Support Automated DEVS Generation and Reuse
Overall Processing

| A | B | C | D | E | F | G
|---|---|---|---|---|---|---
| DSR ID | Requirement Description | SES/NavRepresentation | FDDEVSPresentation | Comments and Questions | Standards |
| 1366 | The IADM shall be capable of inhibiting \( (\leq \text{Global Static C2 Rule Set}) \) automatic correlation of a composite and TDL remote track where either track has the strength field set to a value greater than 1. From the multi perspective, tracks is made of more than one track, A track can be composite or TDL remote in origin A track has a strength field. The range of track's strength field is double with value \([0,1]\]. | IASM1366: to start passive in passive! IASM1366: when in passive and receive EnableAutoCor go to AutoCorMode! IASM1366: passive in AutoCorMode! | Global Static C2 Rule Set |

Augmented Test Models

Automated

FDDEVVS atomic model

manual

FDDEVVS test models

SES

Pruned ES

SES/XML Of DSR texts
DEVS/SOA Infrastructure: Supports Deployment and Execution of DEVS Models on the Web

- Service Oriented Architecture (SOA) consists of various W3C standards
- Client server framework
- XML Message encapsulated in SOAP wrapper
- Machine-to-machine interoperable interaction over the network based on WSDL interface descriptions

Run Example
Deploying Models: DEVSML and DEVS/SOA
Automated Negotiation Support in Multi-Agent Web Environments

- Domain-independent structure
- Domain-dependent behavior

FD-DEVS
- phases
- message types

SES
- message specializations

FD-DEVS Market Place
- Receive message
- Interpret message
- Send message

storeSpec
- Surveillance Spec
- Select
- Input
- Price
- SE
- Return

Location
- Altitude
- Speed
- Roughness
Analysis-Based Network Data Extraction

Network Data Collection

Use Aspects, Specializations, ... and Pragmatic Frame to develop System Entity Structure

SES for Network Data

pruning

SES for Throughput Analysis

SES for Protocol Analysis

SES for Intrusion Detection
Applications

• Natural language capture of high level information technology systems requirements
• Automated generation of FDDEVS kernel DEVSJAVA/C++ models for distributed real-time net-centric IT systems testing
• Development of web service workflows using DEVS/SOA
• Network Traffic data capture, focused extraction, and model generation for exercising IT systems e.g., intrusion detection
Conclusions

• DEVS and SES provide a framework based on Systems Theory for Web-Centric M&S environments
• Supports integrated development and testing
• DEVS standard supports sharable models and repository reuse on the Service Oriented Architecture
• Provides a basis for achieving higher levels of interoperability – can work with HLA or not: DEVS/SOA provides a SOA implementation independent of HLA
• The framework supports development of generic tools which in turn support a wide array of web service domain specific specializations and applications
Books and Web Links

devsworld.org   acims.arizona.edu   Rtsync.com