An Introduction to Modeling and Simulation with DEVS

Gabriel A. Wainer
Department of Systems and Computer Engineering
Carleton University.
Ottawa, ON. Canada.
http://www.sce.carleton.ca/faculty/wainer
Outline

- Problem characterization
- DEVS formalism
- The CD++ tool
- Modeling complex systems using DEVS
- Examples of application

Some of the slides here presented are part of Prof. B. Zeigler’s collection (with permission!)
http://www.acims.arizona.edu
Motivation

- Analysis of complex natural/artificial real systems.

- Continuous systems analysis
  - Different mathematical formalisms
  - Simulation: solutions to particular problems under certain experimental conditions of interest

- Classical methods for continuous systems simulation
  - Based on numerical approximation
  - Require time discretization
  - Inefficient in terms of execution times
  - Complex composition; difficulties in integration, multiresolution models
Evolution in simulation technology

- Reduced cost of modern computers
- Enhanced tools
- Statistical packages; application libraries
- Ease to use, flexibility
- Ease of analysis tasks
- Parallel/Distributed systems
- Enhanced visualization tools
- Standards (graphics, runtime support, distributed software)
Discrete-Event M&S

- Based on programming languages (difficult to test, maintain, verify).
- Beginning ’70s: research on M&S methodologies
- Improvement of development task
- Focus in reuse, ease of modeling, development cost reductions
## DE Modeling and Simulation

<table>
<thead>
<tr>
<th>Example</th>
<th>Safeness, Liveness</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model type</td>
<td>Untimed DES model</td>
<td>Timed DES model</td>
</tr>
<tr>
<td>Required information</td>
<td>State sequence</td>
<td>Timed state sequence</td>
</tr>
</tbody>
</table>

### Behavioral analysis
- (Func. Veri/logical analysis)

### Non-behavioral analysis
- (Performance analysis)

<table>
<thead>
<tr>
<th>Logic base</th>
<th>Temporal Logic</th>
<th>DEVS formalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process algebra</td>
<td>CSP</td>
<td>Generlized Semi-Markov process</td>
</tr>
<tr>
<td></td>
<td>CCS</td>
<td>Min-Max algebra</td>
</tr>
<tr>
<td>Set/Bag theory</td>
<td>FSM</td>
<td>Timed FSM</td>
</tr>
<tr>
<td></td>
<td>Petri net Automata</td>
<td>Timed-PN</td>
</tr>
</tbody>
</table>

(Prof. T. G. Kim, KAIST, Korea)
Separation of concerns in DEVS

Experimental Frame

Real World

Data: Input/output relation pairs

Conditions under which the system is experimented with/observed

Each entity formalized as a Mathematical Dynamic System (mathematical manipulations to prove system properties)

Device for executing model

Simulator

Model

Structure generating behavior claimed to represent real world

Modeling relation

Simulation relation
Current needs

- **Interoperability:**
  - computer-based and non-computer-based systems
    - support a wide range of models and simulations
  - hybrid interoperability

- **Reuse:**
  - model and simulation reuse (computer-based and otherwise)
    - centralized and distributed data and model repositories

- **Performance:**
  - Computational (local to each simulation)
  - Communication (among multiple simulations)
Current practices

• Ad-hoc techniques, ignorance of previous recommendations for software engineering.

• Tendency to encapsulate models/simulators/experimental frames into tightly coupled packages, (written in programming languages such as Fortran, C/C++, Java).

• Difficulties: testing, maintainability of the applications, integration, software reuse.

• Relatively few examples of storing previously developed simulation infrastructure commodities such that they can be adapted to developing interoperability test requirements.
DEVS M&S methodology

• DEVS can be used to solve the previously mentioned issues:
  – Interoperability and reuse
  – Hybrid systems definition
  – Engineering-based approach
  – Facilities for automated tasks
  – Reduced life cycles
  – High performance/distributed simulation
The DEVS M&S Framework

• DEVS = Discrete Event System Specification
• Formal M&S framework
• Supports full range of dynamic system representation capability
• Supports hierarchical, modular model development

(Zeigler, 1976/84/90/00)
The DEVS M&S Framework

• Separates Modeling from Simulation
• Derived from Generic Dynamic Systems Formalism
  – Includes Continuous and Discrete Time Systems
• Provides Well Defined Coupling of Components
• Supports
  – Hierarchical Construction
  – Stand Alone Testing
  – Repository Reuse
• Enables Provably Correct, Efficient, Event-Based, Distributed Simulation
A Layered view on M&S

- Applications
- Models
- Simulators (single/multi CPU/RT)
- Middleware/OS (Corba/HLA/P2P; Windows/Linux/RTOS…)
- Hardware (PCs/Clusters of PC/HW boards…)

Slide 13
A Layered view on M&S

Applications

Models

Simulators (single/multi CPU/RT)

Middleware/OS (Corba/HLA/P2P; Windows/Linux/RTOS…)

Hardware (PCs/Clusters of PC/HW boards…)
Advantages of DEVS

- Models/Simulators/EF: distinct entities with their own software representations.

- Simulators can perform single host, distributed and real-time execution as needed (DEVS simulators over various middleware such as MPI, HLA, CORBA, etc.).

- Experimental frames appropriate to a model distinctly identified; easier for potential users of a model to uncover objectives and assumptions that went into its creation.

- Models/frames developed systematically for interoperability

- Repositories of models and frames created and maintained (components for constructing new models). Models/frames stored in repositories with information to enable reuse.
DEVS Toolkits

- ADEVS (University of Arizona)
- CD++ (Carleton University)
- DEVS/HLA (ACIMS)
- DEVSJAVA (ACIMS)
- GALATEA (USB – Venezuela)
- GDEVS (Aix-Marseille III, France)
- JDEVS (Université de Corse - France)
- PyDEVS (McGill)
- PowerDEVS (University of Rosario, Argentina)
- SimBeams (University of Linz – Austria)
- New efforts in China, France, Portugal, Spain, Russia.
**DEVS Formalism (cont.)**

- Discrete-Event formalism: time advances using a continuous time base.
- Basic models that can be coupled to build complex simulations.
- Abstract simulation mechanism

**Atomic Models:**

\[
M = < X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, D >.
\]

**Coupled Models:**

\[
CM = < X, Y, D, \{M_i\}, EIC, EOC, IC, \text{select} >
\]
DEVS atomic models semantics

\[\lambda (s)\]

\[s' = \delta_{\text{ext}}(s,e,x)\]

\[\delta_{\text{int}}(s)\]

\[\delta_{\text{int}}(s) = D(s)\]

\[\delta_{\text{ext}}(s,e,x) = \delta_{\text{ext}}(s)\]

\[\text{DEVS} = < X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, D, \lambda >\]
DEVS atomic models semantics

\[ s' = \delta_{\text{ext}}(s,e,x) \]

\[ \lambda(s) \]

\[ s' = \delta_{\text{int}}(s) \]

\[ D(s) \]

\[ \delta_{\text{int}}, \delta_{\text{ext}}, D, \lambda \]

\[ \text{DEVS} = < X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, D, \lambda > \]
Coupled Models

Components
couplings
Internal Couplings
External Input Couplings
External Output Couplings

start

start

generator (genr)

out

repair
shop

repaired

faulty

sent

report

finished

transducer (transd)

out

stop

dotted line
Closure Under Coupling

Every DEVS coupled model has a DEVS Basic equivalent
The CD++ toolkit

- Basic tool following DEVS formalism.
- Extension to include Cell-DEVS models.
- High level specification language for model definition.
CD++ simulator

Independent simulation mechanisms
(“Abstract” simulator)
  . Hierarchical
  . Flat
  . Distributed/Parallel
  . Real-Time
Auto-Factory DEVS model
DEVS Graphs Modeling environment
Engine Assembly Atomic

Model &EngineAssem::externalFunction( const ExternalMessage &msg ) {
  if( msg.port() == in_piston ) {  // parts received one by one
    elements_piston.push_back( 1 ) ;
  if( elements_piston.size() == 1 && elements_engineBody.size()>=1)
  holdIn( active, manufacturingTime );
  for(int i=2;i<=msg.value;i++)  //pushback if more than 1 received
    elements_piston.push_back( 1 ) ;
  if( msg.port() == in_engineBody ) { ...}
}

Model &EngineAssem::internalFunction( const InternalMessage & ) { 
  passivate();
}

Model &EngineAssem::outputFunction( const InternalMessage &msg ) {
  sendOutput( msg.time(), out, elements.front());
}
Auto Factory execution

X/00:000/top/in/2 to chassis
X/00:000/top/in/2 to body
X/00:000/top/in/2 to trans
X/00:000/top/in/2 to enginesubfact
D/00:000/chassis/02:000 to top
D/00:000/body/02:000 to top
D/00:000/trans/02:000 to top
X/00:000/enginesubfact/ in/2 to piston
X/00:000/enginesubfact/ in/2 to enginebody ...
Y/02:000/chassis/out/1 to top
D/02:000/chassis/... to top
X/02:000/top/done/1 to chassis
X/02:000/top/in_chassis/1 to finalass ...
*/02:000/top to enginesubfact
*/02:000/enginesubfact to enginebody
Y/02:000/enginebody/out/1 to enginesubfact
D/02:000/enginebody/... to enginesubfact
X/02:000/enginesubfact/done/1 to enginebody
X/02:000/in_enginebody/1 to engineassem
D/02:000/enginebody/02:000 to enginesubfact
D/02:000/engineassem/02:000 to enginesubfact ...
Auto Factory
DEVS Success Stories

- Prototyping and testing environment for embedded system design (Schulz, S.; Rozenblit, J.W.; Buchenrieder, K.; Mrva, M.)
- Urban traffic models (Lee, J.K.; Lee, J-J.; Chi, S.D.; et al.)
- Watershed Modeling (Chiari, F. et al.)
- Decision support tool for an intermodal container terminal (Gambardella, L.M.; Rizzoli, A.E.; Zaffalon, M.)
- Forecast development of Caulerpa taxifolia, an invasive tropical alga (Hill, D.; Thibault, T.; Coquillard, P.)
DEVS Success Stories

- Supply chain applications (Kim, D.; Cao H.; Buckley S.J.)
- Solar electric system (Filippi, J-B.; Chiari, F.; Bisgambiglia, P.)
- M&S activities at the Army base of Fort Wachuka, AZ (B. Zeigler, J. Nutaro et al.)
- Representation of hardware models developed with heterogeneous languages (Kim, J-K.; Kim, Y.G.; Kim, T.G.)
- DEVS/HLA Research funded by DARPA received Honorable Mention in 1999 DMSO Awards
DEVS Bus Concept

Discrete Event Formalisms

Discrete Time Systems

Diff Eq. Systems

DEVS

message

HLA

RTI
Joint MEASURE Overview

- Scenario Specification
- Runtime Visualization/Animation
- Data Analysis
UA/Lockheed distributed experimentation

**JM:**
- Detailed Surface Ship Models
- Sub/Surface Enemy Assets

**Medusa:**
- Hi Fidelity Radar / Weapon Scheduling

**Space Manager and Logger:**
- Pragmatic Event Cue Emission Propagation (with acoustics)

**JM:**
- Space Based Sensors
- Space Based Communication
- Land/Air Enemy Assets

**Space Manager and Logger:**
- Pragmatic Event Cue Emission Propagation

**DEVS/HLA**
- quantization
- predictive filtering
- GIS/aggregation

**LMGES -- NJ**

**LMMS -- CA**

*Slide 34*
## Component Model Reuse Matrix

<table>
<thead>
<tr>
<th>Project</th>
<th>Model</th>
<th>Critical Mobile Target</th>
<th>Global Positioning System III</th>
<th>Arsenal Ship</th>
<th>Coast Guard Deep Water</th>
<th>Space Operations Vehicle</th>
<th>Common Aero Vehicle</th>
<th>Joint Composite Tracking Network</th>
<th>Integrated System Center</th>
<th>Space Based Laser</th>
<th>Space Based Discrimination</th>
<th>Missile Defense (Theater / National)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radar Model</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>IR Sensor Model</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Missile Model</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Laser Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Comm. Model</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Command Control Model</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Earth &amp; Terrain Model</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Weather Model</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Waypoint &amp; Heading Nav Model</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Orbital Propagate Model</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ballistic Trajectory Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

*Note: The table shows the reuse of models across different projects.*

*Slide 35*

Reported gains in development times thanks to the use of DEVS
Sachem = large-scale monitor/diagnose control system for blast furnace operation

Usinor -- world’s largest producer of steel products,

Problems for conventional control and AI:
  - Experts’ perception knowledge: implicit
  - Reasoning of a control process expert: difficult to model.
  - Lack of models for blast furnace dynamics

Solution:
  - time-based perception and discrete event processing for dealing with complex dynamical systems
DEVS framework for control of steel production

Large Scale:
- Conceptual model contains 25,000 objects for 33 goals, 27 tasks, etc.
- Approximately 400,000 lines of code.
- 14 man-years: 6 knowledge engineers and 12 experts

One advantage of DEVS is **compactness**: high reduction in data volume

Effective analysis and control of the behavior of blast furnaces at high resolution
Examples of Application

- Models of an **Intel 8086** CPU and **DSP** processors (VoIP).

- Simple **Digital systems** (vending machine, alarm clock, plant controller, robot path finder). Interpreter of **VHDL** and nVHDL.


- Computer **communication**: routing protocols for LANs, IP6, client/server models, simple protocols.

- **Physical**: excitable media, particle collision, flow injection.

- **Geographical/Ecological**: fire spread, plant growth, watershed formation, erosion, ant foraging.

- **Biosystems**: mythocondria, heart tissue, bacteria spread.
$\alpha - 1$ simulated computer
Physical Systems

- Heat Spread

- Surface Tension

- Binary solidification
Fire Spread Modeling

Slide 42

$T_{ig} = 573\, K$

$T_f = 333\, K$

Active

Unburned

Burning

Burned

$t = (t_i, T_a)$
Watershed modeling
Pursuer/evader modeling
Vibrio Parahaemolyticus bacteria

Temperature

Bacteria concentration

Initial | After 1.5 hr | After 4 hrs
Ants following pheromone paths

Ants seeking food

t=1

Sources of food

Ants found pheromone path

t=2

Ants returning to nest

t=3

t=4
Different phases of the algorithm: (a) Configuration of obstacles, (b) Boundary detection, (c) Information for CA Expansion, (d) Optimal collision-free path
Flow Injection Analysis (FIA)

FIA manifold. P: pump; A,B: carrier and reagent lines; L: sample injection; I: injection valve; R: reactor coil; D: flow through detector; W: waste line.

- P pumps carrier solution A into valve I that connects to reactor R
- By turning valve I, sample B is injected into R
- Reactions in R between A and B are sensed by detector D
Heart tissue behavior

- Heart muscle excitable; responds to external stimuli by contracting muscular cells.
- Equations defined by Hodgkin and Huxley
- Every cell reproducing the original equations
- Discrete time
- Discrete event approximation
- G-DEVS, Q-DEVS
Test cases: a heart tissue model

- Automated discretization of the continuous signal
A Watershed model

Excedent water to neighbor \( l^{vs}(t) \)

Surface vegetation

Effective water \( le(t) \)

Accumulated water \( Ac(t) \)

Land absorption \( f(t) \)

Rain \( l(t) \)

Water received from the neighbors \( l^{ve}(t) \)

WSHED - Topology - Time 0

WSHED - Quantum Hys 1.0 - After 10'
Flow Injection Analysis Model

No Quantum, 120ms

Q-DEVS 0.1, 120ms

Quantum Standard 0.7 Dynamic 1 - 0.05, 120ms
ATLAS SW Architecture
Modelling a city section

- 24-line specification
- 1000 lines of CD++ specifications automatically generated
Describing a city section
Defining a city section in MAPS
Exporting to TSC
Visualizing outputs
Modeling AODV routing

- Variant of the classical Lee’s Algorithm.
- $S$: node; $D$: a destination; black cells: dead.
- $S$ broadcasts RREQ message to all its neighbors (wave nodes).
- Wave nodes re-broadcast, and set up a reverse path to the sender.
- The process continues until the message reaches the destination node $D$.
- Shortest path is selected.
Simulation results
Execution results
Internetworking Routing

- 3D Cell-DEVS model
- Plane 1: wireless network, Plane 2: wired.
Maze solving

(1) 

(2) 

(3)
Simulated results

- Creation of a 3D version of the simulation
- Interpreted by the MEL scripts
Path plane
3D Simulation
Advantages of DEVS

• Reduced development times
• Improved testing => higher quality models
• Improved maintainability
• Easy experimentation
• Automated parallel/real-time execution
• Verification/Validation facilities
Difficulties of DEVS

• **Legacy** (current experience of modelers)

• Building DEVS models is **not trivial**
  • Petri Nets, FSA, etc. more successful

• **Training**
  • Differential Equations
  • State machines
  • Programming
Where to go from now

- Bridging the gap between Academic world and actual Application users
  - DEVS ready to take the leap
  - Critical mass of knowledgeable people
  - Large amount of tools/researchers
  - Ready to go from Research to Development
- Standardization of models
- Building libraries/user-friendly environments
- Further research required; open areas.
Standardization at the right level

Objectives:
- Support interoperability at the modeling level
- Support model composability
- Technology independence
- Include all modeling paradigms in wide use
- Research collaboration academia, industry, gov

Legend

DTSS – Discrete Time System Specification
ODESS – Ordinary Differential Equation System Specification
PDESS – Partial Differential Equation System Specification
Concluding remarks

- DEVS formalism: enhanced execution speed, improved model definition, model reuse.
- Hierarchical specifications: multiple levels of abstraction.
- Separation of models/simulators/EF: eases verification.
- Experimental frameworks: building validation tools
- Modeling using CD++: fast learning curve
- Parallel execution of models: enhanced speed
- The variety of models introduced show the possibilities in defining complex systems using Cell-DEVS.
- Incremental development
Current work and a research roadmap

http://www.sce.carleton.ca/faculty/wainer