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

• FOCUS => Newcomers

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
Separation of concerns in DEVS

- Experimental Frame
  - Data: Input/output relation pairs
  - Conditions under which the system is experimented with/observed
- Real World
- Modeling relation
- Simulator
- Simulation relation
- Model
- Structure generating behavior claimed to represent real world

Each entity formalized as a Mathematical Dynamic System (mathematical manipulations to prove system properties)
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
- Separates Modeling from Simulation
- Supports full range of *dynamic system* representation capability
- Supports hierarchical, modular model development
- Provides Well Defined Coupling of Components
- Supports
  - Hierarchical Construction
  - Stand Alone Testing
  - Repository Reuse

(Zeigler, 1976/84/90/00)
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 11
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 12
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)
- **James** (University of Rostock, Germany)
- **JDEVS** (Université de Corse - France)
- **PowerDEVS** (University of Rosario, Argentina)
- **SimBeams** (University of Linz – Austria)
- **SmallDEVS** (University of Brno, Czech Republic)
- **VLE** (Université du Litoral - France)
- 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:

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

Coupled Models:

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

\[ \lambda (s) \]

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

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

\[ D(s) \]

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

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

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

\[ D(s) = \lambda(s) \]

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

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

\[ \text{DEVS} = \langle X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, D, \lambda \rangle \]
Dynamic behavior
Components couplings
Internal Couplings
External Input Couplings
External Output Couplings

Coupled Models

- generator (genr)
- transducer (transd)
- repair shop
- repair
- report
- start
- stop
- sent
- finished
- out
- faulty
- repaired
- out
- report
Closure Under Coupling

DN
\(< X, Y, D, \{M_i\}, \{I_i\}, \{Z_{ij}\}>\)

DEVS
\(< X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \delta_{\text{con}}, ta, \lambda >\)

Every DEVS coupled model has a DEVS Basic equivalent

\(\text{Every DEVS coupled model has a DEVS Basic equivalent}\)
Quantized DEVS (QDEVS)

- Continuous signal represented by crossing of an equal spaced set of boundaries, separated by a *quantum* size.

- Check for boundary crossing for every change in the model.

- Outputs generated only when a crossing occurs.

- Substantial reduction of the message updates frequency.

![Diagram showing quantization and boundary crossing](image-url)
Cell-DEVS models

- Discrete-Events cell spaces
- Cells: atomic models. Automated coupling.
- Asynchronous execution using explicit delay functions
- Abstract simulation mechanism.
Cell-DEVS Atomic Models

Transport Delay

Inertial Delay

\[ \text{TDC} = < X, Y, \theta, N, d, \tau, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, D > \]

- N inputs to a given cell
- Local computing function
- Inertial or Transport delays
- Outputs only if the cell state changes
Coupled Cell-DEVS

GCC = < Xlist, Ylist, X, Y, n, \{t_1, ..., t_n\}, N, C, B, Z >
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

The diagram illustrates the DEVS (Discrete Event System Specification) model of an automobile factory. The model includes subfactories for Chassis, Body, Transmission Case, Engine, Piston Line, Engine Body Line, and the Final Assembly Subfactory. The Production Plan is the input, and the output is Automobiles.
DEVS Graphs Modeling environment
Engine Assembly Atomic

Model EngineAssem::EngineAssem(const string &name):Atomic(name),
in_piston(addInputPort("in_piston")), in_engineBody(addInputPort("in_engineBody")), done(addInputPort("done")), out(addOutputPort("out")),
manufacturingTime(0, 0, 10, 0) {} // Model constructor

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.)
• Intrusion Detection Systems (Cho, T.H.; Kim, H.J.)
• Depot Operations Modeling (B. Zeigler et al. U.S. Air Force)
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 JITC, 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

- DEVS
- message
- HLA

Discrete Time Systems

- DEVS
- message
- HLA

Diff Eq. Systems

- DEVS
- message
- HLA

RTI
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 36
<table>
<thead>
<tr>
<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>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</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>
</tr>
<tr>
<td>Missile Model</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>Comm. Model</td>
<td>X</td>
<td>X</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>X</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>X</td>
<td>X</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></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>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Orbital Propagate Model</td>
<td>X</td>
<td>X</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>X</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>

Reported gains in development times thanks to the use of DEVS
**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**
α-1 simulated computer
Physical Systems

- Heat Spread
- Surface Tension

- Binary solidification
Fire Spread Modeling

\[ T_g = 573 \text{ K} \]
\[ T_f = 333 \text{ K} \]

Active, Unburned, Burning, Burned
Watershed modeling
Metabolic pathways: Glycolisis

- Sequence of reactions used by cells to metabolize glucose
- Role: to produce energy
- Glycolysis generates about 15% of energy produced by aerobic respiration
- A sequence of ten reactions
- Converts one glucose molecule into two pyruvate molecules
- Produces NADH and ATP.
- Specific enzymes control each of the different reactions.
Step 1 Atomic Model

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

\[ \delta_{\text{int}}, \delta_{\text{ext}}, \tau_a \text{ and } \lambda \text{ using CD++ implementation.} \]

\[
\text{Model \\& Step1::externalFunction} \\
\quad (\text{const ExternalMessage} &\text{msg}) \{ \\
\quad \text{if} (\text{msg.port()} == \text{glucose}) \{ \\
\quad \quad \text{glucosec} = \text{glucosec} + \text{msg.value}(); \\
\quad \quad \text{if} ((\text{atpc} > 0) \&\& (\text{ifhex} == \text{true} )) \\
\quad \quad \quad \text{holdIn}(active, \text{Prep_Gluc}); \\
\quad \}\text{else if}(\text{msg.port()} == \text{ATPi}) \{ \\
\quad \quad \text{atpc} = \text{atpc} + \text{msg.value}(); \\
\quad \quad \text{if} ((\text{glucosec}>0) \&\& (\text{ifhex}==\text{true} )) \\
\quad \quad \quad \text{holdIn}(active, \text{Prep_ATPi}); \\
\quad \}\text{else if}(\text{msg.port()} == \text{hexokinase}) \{ \\
\quad \quad \text{ifhex} = \text{true} ; \\
\quad \quad \text{if} ((\text{glucosec} > 0) \&\& (\text{atpc} > 0)) \\
\quad \quad \quad \text{holdIn}(active, \text{Prep_Hexo}); \\
\quad \}\}
\]
At time 40:00: after four glucose molecules entered the cell; four more outputs of each of the ADP, glucose_6_phosphate, and H molecules.
Coupled Animation of Glycolysis
Glycolisis 3D visual results
After 38 times steps, the immune cells have cleared away proliferative cells on the north side of the tumor.

After 64 times steps, the tumor overwhelms the immune system.
Pursuer/evader modeling
Vibrio Parahaemolyticus bacteria

Temperature

Bacteria concentration

Initial
After 1.5 hr
After 4 hrs
Ants seeking food

Sources of food

Ants found pheromone path

Ants returning to nest

Ants following pheromone paths
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

- Rain \( l(t) \)
- Effective water \( le(t) \)
- Surface vegetation
- Accumulated water \( Ac(t) \)
- Excedent water to neighbor \( lvs(t) \)
- Water received from the neighbors \( lve(t) \)
- Land absorption \( f(t) \)
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.
Network Prototyping

- Real time simulation on embedded Linux microcontrollers
- Rapid design and testing potential network devices
Modelica/CD++

**model circuit**
V(V=10, width=50, period=2.5);
Modelica.Electrical.Analog.Basic.Resistor R1(R=0.001);
Modelica.Electrical.Analog.Basic.Inductor I1(L=500);
Modelica.Electrical.Analog.Basic.Inductor I2(L=2000);
Modelica.Electrical.Analog.Basic.Capacitor C(C=10);
Modelica.Electrical.Analog.Basic.Resistor R2(R=1000);

**equation**
connect(V.p, R1.p);
connect(R1.n, I1.p);
connect(R1.n, I2.p);
connect(I2.n, C.p);
connect(I2.n, R2.p);
connect(C.n, I1.n);
connect(R2.n, C.n);
connect(I1.n, V.n);
connect(V.n, Gnd.p);

**end circuit;**
M/CD++ Execution Example

Capacitor.v curves comparison

Time (sec) [0.00 to 12.00]
Capacitor.v [0.00E+00 to 1.60E-02]

MCD++ interpolated by Dymola
MCD++
M/CD++ Execution Example

Inductor1.i curves comparison

Time (sec)

Inductor1.i

MCD++ interpolated by Dymola

MCD++
Sample Model Execution

- Multiple model controller allowed to operate as designed, and switch among plant identifying models

- Controller was able to find it and use its parameters

- Error existed only at the period coinciding w/each jump in plant parameters

- Only at time 355 did a false model switch occur (due to two models having almost zero error)
Incremental Prototyping

- We show how to develop incrementally a model based on simple components.

- The application executes in a simulated environment (i.e., all of the components remain executing in a virtual world).

- Simple model of an elevator with both hardware and simulated components.
An elevator control system

INPUTS

<table>
<thead>
<tr>
<th>Time</th>
<th>Deadline</th>
<th>In-port</th>
<th>Out-Port</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:11:500</td>
<td>00:11:700</td>
<td>btn_3</td>
<td>led3</td>
<td>1</td>
</tr>
<tr>
<td>00:14:600</td>
<td>00:14:800</td>
<td>sensor_2</td>
<td>flr_display</td>
<td>1</td>
</tr>
<tr>
<td>00:19:500</td>
<td>00:19:700</td>
<td>sensor_3</td>
<td>flr_display</td>
<td>1</td>
</tr>
<tr>
<td>00:25:100</td>
<td>00:25:300</td>
<td>btn_4</td>
<td>led4</td>
<td>1</td>
</tr>
<tr>
<td>00:30:000</td>
<td>00:30:200</td>
<td>sensor_4</td>
<td>flr_display</td>
<td>1</td>
</tr>
<tr>
<td>00:36:300</td>
<td>00:36:500</td>
<td>btn_3</td>
<td>led3</td>
<td>1</td>
</tr>
<tr>
<td>00:41:400</td>
<td>00:41:600</td>
<td>sensor_3</td>
<td>flr_display</td>
<td>1</td>
</tr>
</tbody>
</table>

OUTPUTS

<table>
<thead>
<tr>
<th>Time</th>
<th>Deadline</th>
<th>Out-port</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:11:510</td>
<td>00:11:700</td>
<td>led3</td>
<td>1</td>
</tr>
<tr>
<td>00:11:510</td>
<td></td>
<td>dir_display</td>
<td>1</td>
</tr>
<tr>
<td>00:14:610</td>
<td>00:14:800</td>
<td>flr_display</td>
<td>2</td>
</tr>
<tr>
<td>00:19:510</td>
<td>00:19:700</td>
<td>led3</td>
<td>0</td>
</tr>
<tr>
<td>00:19:510</td>
<td></td>
<td>flr_display</td>
<td>3</td>
</tr>
<tr>
<td>00:19:510</td>
<td></td>
<td>dir_display</td>
<td>0</td>
</tr>
<tr>
<td>00:25:110</td>
<td>00:25:300</td>
<td>led4</td>
<td>1</td>
</tr>
<tr>
<td>00:25:110</td>
<td></td>
<td>dir_display</td>
<td>1</td>
</tr>
</tbody>
</table>
Replacing components

components: elevBox   ec@ECU   dis@Display

in : button_1  button_2  button_3  button_4
out : flr_display
link : button_1 button_1@ec
link : button_2 button_2@ec
(...)
link : sensor_1@elevBox sensor_1@ec
link : sensor_2@elevBox sensor_2@ec
(...)
link : floor_disp@ec flr_display@dis
link : floor_disp@ec floor_disp
link : dir_disp@ec dir_display@dis
link : led_1@ec led_1@dis
(...)
[elevBox]
components: sb@SensorController  eng@Engine
in : activate direction
out : sensor_1 sensor_2 sensor_3 sensor_4
link : activate activate@eng
link : direction direction@eng
link : sensor_1@sb sensor_1
(...)
link : current_floor@eng sensor_triggered@sb
(...)

Time          Out-por      Value
00:08:170      flr_display  2
00:19:540      flr_display  1
00:30:130      flr_display  2
00:35:140      flr_display  3
00:40:150      flr_display  4
00:58:290      flr_display  3
Replacing components

<table>
<thead>
<tr>
<th>Time</th>
<th>Port</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:06:120</td>
<td>direction</td>
<td>1</td>
</tr>
<tr>
<td>00:06:130</td>
<td>activate</td>
<td>1</td>
</tr>
<tr>
<td>00:15:930</td>
<td>activate</td>
<td>0</td>
</tr>
<tr>
<td>00:56:800</td>
<td>direction</td>
<td>2</td>
</tr>
<tr>
<td>00:56:810</td>
<td>activate</td>
<td>1</td>
</tr>
<tr>
<td>01:01:130</td>
<td>activate</td>
<td>0</td>
</tr>
<tr>
<td>01:22:710</td>
<td>direction</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Out-port</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:06:130</td>
<td>result</td>
<td>1</td>
</tr>
<tr>
<td>00:15:930</td>
<td>result</td>
<td>0</td>
</tr>
<tr>
<td>00:56:810</td>
<td>result</td>
<td>2</td>
</tr>
<tr>
<td>01:01:130</td>
<td>result</td>
<td>0</td>
</tr>
</tbody>
</table>

components: eng@Engine
in : activate_in direction_in
out : result
link : activate_in activate@eng
link : direction_in direction@eng
Building a Robot Controller
Integration Tests
Integration Tests
Parallel CD++

- Execute DEVS models in parallel
- Layered architecture based on different middleware
  - Expansion to RTI: few lines of code

Hardware: Cluster of Processors/Myrinet
Middleware: Warped/MPI
Parallel Simulation Engine
CD++ Models
Partitioned Fire Model test

- Fire model (1-machine)

![Diagram](image-url)

**Total Execution Time (Fire Model- 1&2 Machines)**

- 1 Machine
- 2 Machines (Internet)
- 2 Machines (UCLP)
Fire Spreading Simulation Mashup

- Finalist at IEEE Services Computing Contest
- Service integrates prediction of forest fires, weather data and Google Maps
Select location and check weather

Use the fields and the map above to select the country and the town where to run the simulation. If the town does not appear or is not supported, please do a manual search with the google map.

Fredericton, N. B., Canada
Time: Aug 12, 2008 - 04:33 PM EDT / 2008.08.12 2033 UTC
Wind: from the NE (050 degrees) at 3 MPH (3 KT)
Visibility: 12 mile(s):0
SkyConditions: mostly cloudy
Temperature: 64 F (18 C)
Mashup (Google Maps)
CD++ Visualization Engines
1. Change geometry, color and size of the nodes
2. Navigation
3. Edit individual node
DEVSView

- Visual models extracted from CD++ simulation log file; visual state machines defined using the DEVSView user interface.
Simulated results

- Creation of a 3D version of the simulation
- Interpreted by the MEL scripts
Evacuation Results
Evacuation Model
3D Visualization Scenario
Boulevard St. Laurent (MTL)

http://www.cims.carleton.ca/pose
Evacuation in St. Laurent Blvd.
SAT Evacuation Animation
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
Further Information

http://cell-devs.sce.carleton.ca