V-Lab® – A Soft Computing Based Virtual Laboratory for Autonomous Agents Simulation and Implementation

Mo Jamshidi, Ph.D., DEgr., Dr. H.C.
F-IEEE, F-ASME, F-AAAS, F-NYAS, F-HAE, F-TWAS
Regents Professor, Electrical and Computer Engr. Department and
Director, Autonomous Control Engineering (ACE) Center
University of New Mexico, Albuquerque

http://ace.unm.edu  http://valb.unm.edu  moj@wacong.org
OUTLINE

• Introduction to V-Lab®

• Discrete-EVent simulation Specifications (DEVS)

• Intelligent DEVS … I-DEVS

• Applications to Multi-agent Systems

• Simulation and Experiment Movies
V-Lab® is ...

... a **NASA Research Announcement** (NRA), **Cross-Enterprise** Grant with NASA Ames Research Center and a joint effort between UNM ACE Center, JPL, in cooperation with the Arizona Integrate Modeling and Simulation Center (U Arizona and Arizona State U) Dr. **Edward Tunstel, Jr.** of JPL is one of the 3 Co-PIs on the grant.
V-Lab®’s potential role for NASA
Mission Context for Mobility Systems R&D

Airborne Systems
- Mars Airplane
- Solar Montgolfiere
- Titan Blimp
- Jupiter Balloon

Surface Systems
- Sojourner
- MER
- Nanorovers
- Inflatables
- Science Outpost

Sub-surface Systems
- Sub-surface Explorer
- Mini-corer
- US Drill/Corer
- Deep Drills
- Cryobot
- Hydrobot

State of the Art (SOA)
SOA + 5 years
SOA + 10 years
Simulation Design

Qualities

■ **Layered Architecture**
  ■ Breaks up the entire simulation into layers, each with a distinct purpose and function

■ **Object Modularity**
  ■ Functional components of the individual layers should be broken up into distinct objects to allow for:
    • Maintainability – Changing a single object without modifying entire simulation.
    • Reuse – Distinct objects can be reused in different simulations
    • Object Hierarchy – Provide a structure for the composition and communication of objects

■ **Our Solution – V-Lab®**
V-Lab® Environment Layers

- **Hardware Networking**
  - Networking foundation for inter-machine communication

- **Middleware**
  - Software environment for inter-process communication (CORBA, HLA, Sockets, etc.)

- **I-DEVS**
  - Library of tools for soft computing in DEVS formalism

- **V-Lab®**
  - Organizational structure for DEVS objects.
  - Management objects to control time and message flow in multi-agent systems
Inside Layered Architecture
DEVS

- Provides C++ and Java classes for object design

**Hierarchical structure of objects**

- Two Types of Objects
  - **Coupled (A-BC, B-C)**
    - Contains other objects
  - Atomic (A, B, C)
    - Basic functionality to be used by coupled objects

Diagram:

- Hierarchical tree:
  - A-BC
  - A
  - B-C
  - B
  - C

- Coupling Relation:
  - A-BC
  - A
  - B-C
  - B
  - C
DEVS execution

- **Create Models**
- Determine imminent models
  - Imminent models send messages to output ports
  - **Imminent models call Internal Transition function**
  - Other models receive external messages
  - Other models call External Transition function
- **Repeat until no models can be imminent**
IDEVS ... Intelligent DEVS

DEVS

Coupled Models

Atomic Models

Soft Computing

Discrete Events
Fuzzy-DEVS

Object: Implement various fuzzy functions and operations within a DEVS Java environment.

Example – fuzzy inferencing

A DEVS model for a typical fuzzy rule:

IF x is A OR y is B THEN z is C
Fuzzy-DEVS, Cont’d.

Example of using fuzzy-DEVS to control an inverted pendulum

a) Membership functions for fuzzy-DEVS

b) The fuzzy-DEVS controller

c) The close-loop system

D) The output of the system ($\theta$)
Fuzzy Logic Controller

- Fuzzifier
- Inference Engine
- Defuzzilier

Crisp Values → Fuzzifier → Inference Engine → Defuzzilier → Crisp Values

- Membership Functions
- Fuzzy (IF THEN) Rules
- Fuzzy Variables
- Linguistic Variables
Devs-Fuzzy Logic

A typical Mamdani rule can be composed as follows:

If $x_1$ is $A_1^i$ AND $x_2$ is $A_2^i$ THEN $y^i$ is $B^i$, for $i = 1, 2, ..., l$

DEVS primitives to compose a Mamdani rule:
IF \(\theta\) is POS AND \(\theta_{dot}\) is POS THEN force is NEG
GA’s Basic Cycle

- New Population
- Recombination
- Selection
- Old Population
- Evaluation

Crossover Mutation

A binary chromosome:

0 1 0 0 1 1 0 1 0 1 0 0
GA-DEVS, Cont’d.

GA coupled model

GA-DEVS implementation inside V-Lab®
A simulation window of GA-DEVS
Average and maximum fitness versus generation for an example problem
What is a Stochastic Learning Automata?

- SLA is a sequential machine.
- Given a finite number of actions that can be performed in a random environment, when a specific action is taken place the environment provides a random response which is either favorable or unfavorable.
- The objective in the design of the automaton is to determine how the choice of the action at any stage should be guided by past actions and responses.
Why SLA?

- Successful in solving complex, highly nonlinear, uncertain, incomplete or non-stationary problems
- Need no knowledge of the model of the process to be controlled.
- Need no analytical knowledge of the function to be optimized.

SLA is …

- Connected in feedback loop to the environment
- Is a sequential machine characterized by a set of
  - internal states
  - input actions
  - state probability distributions
  - reinforcement scheme
  - output function

- The probability distribution is adjusted using reinforcement scheme to achieve the desired objective
Stochastic Learning Automata – Reinforcement Learning

Stochastic Learning Automaton

G(.)

Update Action Probabilities

F(w(n),s(n))

y(n)

s(n)
Stochastic Learning Automaton – A statistical Learning approach to learning ...

Schematic of SLA inside the DEVS Environment
Neural Networks – A neural-based approach to learning ...

A perceptron structure inside DEVS
Neural Networks – Logical connectives “AND”, “OR”, and “XOR” are implemented in IDEVS

Number of training data errors versus epochs
(X, training epochs; Y, number of training data errors)
Cover – Modeling & Simulation Society International Magazine, 2003
V-Lab®

- Uses the **I-DEVS** structure for creating objects
- Organizes objects into 6 different categories
  - SimEnv/SimMan
  - Terrain Models
  - Agent Models
  - Physics Models
  - Control Models
  - Dynamic Models
- Provides objects for the time and message flow of simulation
SimMan and SimEnv

- **SimEnv**
  - Highest leveled coupled model
  - Instatiates all other models
  - Houses all other models
  - Couples all other models together

- **SimMan**
  - Message liaison providing indirection between all other models
  - Controls flow of time for the simulation
High-Level Models

- **Physics**
  - Model physical phenomena.
  - Possibly a differential equation solver and set of differential equations

- **Terrain**
  - Information about the layout of the environment

- **Dynamic**
  - Analyze agent’s actuator state to make environmental changes, e.g. agent’s position and velocity.
Simulation execution is iteration through a 5 phase cycle

- **Phase 1**
  - Check for termination conditions
- **Phase 2**
  - Update Agent Sensors
- **Phase 3**
  - Control Algorithms update Agent Actuators
- **Phase 4**
  - Wait for all actuators to change state
- **Phase 5**
  - Dynamic models update agents and environment
High-Level Models

- **Control**
  - Algorithms that control the actuators of an agent model

- **Agent**
  - Represent the agents in the simulation
  - Contain Sensor models
    - Sensor models contain all information about the external world an agent is aware of
  - Contain Actuator models
    - Actuators contain state information that dynamic models use
Simulation, SLA-DEVS

2 Agents system

Top view of simulation
Simulation, Cont’d ... 2 Agents

(a) 
(b) 
(c) 
(d)
Simulation, Cont’d … 2 Agents

(e) (f)

(g) (h)
A THEME EXAMPLE SIMULATION

Block Diagram of Theme Example
A THEME EXAMPLE

SIMULATION – Rover Dynamics

\[ \begin{align*}
\dot{x} &= \frac{1}{2}(v_r + v_l) \cos \theta \\
\dot{y} &= \frac{1}{2}(v_r + v_l) \sin \theta \\
\dot{\theta} &= \frac{1}{l}(v_r - v_l)
\end{align*} \]
Rule 1: IF distance is ‘Near’, THEN forward velocity is ‘Slow’.
Rule 2: IF distance is ‘Far’, THEN forward velocity is ‘Speedy’.
Rule 3: IF angle is ‘Right’, THEN angular velocity is ‘TurnRight’; forward velocity is ‘Slow’.

etc.
**Infra-red sensors.** ... Each sensor has a unique ID, X_s, Y_s and theta_s, where X_s is the X-position, Y_s is the Y-position and theta_s is the angle of each sensor relative to the rover in X-Y plane.

**GPS sensors** ... An atomic model simply simulates a position sensor, giving out the current position of the rover.

**Compass Sensors** ... Respective atomic models simulate the sensing by converting the angle θ of the rover from radians to degrees.
A THEME EXAMPLE
SIMULATION - Results
A THEME EXAMPLE
SIMULATION – Results 2
• The distributed capability via CORBA is not supported in the latest version of DEVSJAVA® 2.7

• Instead of employing CORBA as middleware, GenDEVS provides distributed simulation with TCP/IP sockets as the middleware.

• The coordinator must wait for each client component to connect/register with it, so it is the user’s responsibility to also start the client components (as processes on different hosts) from somewhere on the network.

• On the server, a thread is spawned for each connecting component as part of the registration process.

• This thread is just a proxy that handles communication between a single client component and the central coordinator/server. (see Figure)
Parts of a distributed simulation with GenDEVS and Sockets
Pushing task:
Cooperative pushing task:
Cooperative Pushing Task
Simulation Results
GA-DEVS Controller (1)

\[
\begin{align*}
\dot{x} &= \frac{1}{2}(v_r + v_l) \cos \theta \\
\dot{y} &= \frac{1}{2}(v_r + v_l) \sin \theta \\
\dot{\theta} &= \frac{1}{l}(v_r - v_l)
\end{align*}
\]

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix} = v \begin{bmatrix}
\cos \theta \\
\sin \theta \\
0
\end{bmatrix} + w \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix}
\]
GA-DEVS Controller (2)

- Target Direction
  - Robot \((x, y, \theta)\)
  - Goal \((x_g, y_g)\)

\[
\theta_g - \theta = \tan^{-1}\left(\frac{y_g - y}{x_g - x}\right)
\]

Goal: left
Goal: front
Goal: behind
Goal: right
GA-DEVS Controller (3)

First simulation sample task using the GA control system, $T=142$ steps
GA-DEVS Controller (4)

Second simulation sample task using the GA control system, T=236 steps
GA-DEVS Controller (5)

- Trajectory of the robot in a task

Trajectory of the robot for first task

Trajectory of the robot for second task
Commercial ROVERS

Two Pioneer II Mobile rovers are used to implement IDEVS-JAVA (V-Lab®).

Movies will show demos at the end.
Experiment Results (DEVS-Fuzzy)
Experiment Results

**GA-DEVS**

- Application: Pioneer 2 mobile robot
- Two experiments of path planning
  - Goal searching task w/o obstacles. **Robot** $(10, 9, -\frac{\pi}{2}) \rightarrow (4, 4, -\frac{\pi}{2}) \rightarrow$ returns to the home point $(10, 9, -\frac{\pi}{2})$ after traveling to the goal point.
Experiment Results (GA-DEVS)

- Application: Pioneer 2 mobile robot

- Two experiments of path planning
  - *Goal searching task w/ obstacles.* **Robot** \((4, 4, -\frac{\pi}{2}) \rightarrow (10, 9, -\frac{\pi}{2})\), avoids obstacle.

Robot task of goal seeking while avoiding obstacle
Conclusions

- **V-Lab®** is still at the beginning of its development
- Three sub-groups are working:
  - **Software**: Responsible for the design and implementation of V-LAB® (from Network to V-LAB®)
  - **Hardware**: Implementation of its tasks, features on industrial or commercial rovers
  - **Theory**: Develop new theories to bridge the gap between Soft Computing (FL, NN, GA, GP, SLA, etc.) and DEVS environment.
- **Future**: SoSE applications of V-Lab® are among future directions.


$\ddot{x} = \int \dot{y} \, dt$, $\vec{v} = \int \frac{\vec{F}}{m} \, dt$
THANK YOU!