DEVS and DEVS Model

Dr. Xiaolin Hu
Outline

• Review of last class
• DEVS introduction
• How DEVS model works
• Simple atomic models (SISO)
• Simple atomic models (with multiple ports)
• Simple coupled models
Event list Scheduling Exercise

• (2) inter-gen-time = 7; service-time = 10

Event: Generate Job
nr-waiting = nr-waiting + 1
schedule a Generate Job in inter-gen-time
if nr-waiting = 1 then
schedule a Process Job in service-time

Event: Process Job
nr-waiting = nr-waiting - 1
if nr-waiting > 0 then
schedule a Process Job in service-time

---

<table>
<thead>
<tr>
<th>Time</th>
<th># wait</th>
<th>Job 1</th>
<th>Job 2</th>
<th>Job 3</th>
<th>Job 4</th>
<th>Job 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Gen_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>Gen_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>Pro_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>Gen_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>Pro_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>Gen_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>2</td>
<td>Gen_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>3</td>
<td>Pro_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Event list Scheduling Exercise

- (1) inter-gen-time = 7; service-time = 5
- (2) inter-gen-time = 7; service-time = 10

Event: Generate_Job
nr-waiting = nr-waiting + 1
schedule a Generate_Job in inter-gen-time
if nr-waiting = 1 then
    schedule a Process_Job in service-time

Event: Process_Job
nr-waiting = nr-waiting - 1
if nr-waiting > 0 then
    schedule a Process_Job in service-time
Discussion of Event Scheduling

Event: Generate_Job
nr-waiting = nr-waiting + 1
schedule a Generate_Job in inter-gen-time
if nr-waiting = 1 then
  schedule a Process_Job in service-time

Event: Process_Job
nr-waiting = nr-waiting - 1
if nr-waiting > 0 then
  schedule a Process_Job in service-time

Break-Ties by: Process_Job then
Generate_Job

• This is a non-modular approach.
  – For example, the Generate_Job event routine may schedules a Process_Job event.
  – Cannot separated the code of different event routines.
• The model and simulator are not separated.
  – An event routine can add/change/delete event in the event list directly.
System Specification:

**DESS**: differential equation
**DTSS**: discrete time
**DEVS**: discrete event

DESS model:
\[
\frac{dq}{dt} = a^*q + bx
\]

DTSS model:
\[
q(t+1) = a^*q(t) + b^*x(t)
\]

Dynamic Systems Framework for Continuous and Discrete Models
Is there a formal way to specify discrete event models?
DEVS Background

• DEVS = Discrete Event System Specification
• Provides formal M&S framework: specification, simulation
• Derived from Mathematical dynamical system theory
• Supports hierarchical, modular composition
• Object oriented implementation
• Supports discrete and continuous paradigms
• Exploits efficient parallel and distributed simulation techniques
• DEVS is a modular modeling approach. A DEVS model does not directly schedule other models’ events.
• DEVS separates a model and its simulator in an explicit way.
  – We focus on the modeling aspect in this class.
• DEVS defines a specification for discrete event models.
SYSTEM

input

state

output
Hierarchical construction
Java Programming – A Holislitic View

• Level 1: know the programming language. Write simple programs.
  – syntax, programming environment…

• Level 2: handle complexity
  – e.g., 1D-array → 2D-array, a program with 2 classes → a program with 20 classes

• Level 3: algorithm design
  – e.g., how to search 1M records in an efficient way

• Level 4: system design
  – e.g., developing a software system, e.g., iCollege, for a given problem

• Level 5: problem solving
  – Formulating a real world problem into a computer science problem and solve it (e.g., using people’s mobile phones to enable earthquake early warning).

The focus of CSC1302 class
You will be able to find a job after reaching this level

Most PhD students work on problems at this level
DEVSJAVA Modeling – A Holistic View

• Level 0: know Java programming

• Level 1: know how DEVS works. Write simple DEVS models (Note: DEVS itself does not depend on JAVA).
  – Semitics of DEVS model, syntax of DEVSJAVA, modeling and simulation environment…

• Level 2: handle complexity
  – e.g., Multiple steps of state transitions, multiple output generation, time tracking, multiple layers of coupled models.

• Level 3: system modeling
  – e.g., developing a complete system model for a given problem.

• Level 4: problem solving
  – Formulating a research problem into a modeling & simulation problem and solve it (e.g., how to use modeling and simulation to study infectious disease spreading or HIV transmission).

The focus of this week and next week

Homework Assignment 1

Homework Assignment 2

Your PhD Research (if you want to use M&S for your research)
DEVS Models

There are two kinds of Models in DEVS

• Atomic Model
• Coupled Model
DEVS Atomic Model

Elements of an atomic model:

• input events
• output events
• state variables
• state transition functions
  – External transition
  – Internal transition
  – Confluent transition
• output function
• time advance function
DEVS Atomic Model Formalism

A Discrete Event System Specification (DEVS) is a structure

\[ M = <X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta> \]

where

- \( X \) is the set of input values.
- \( S \) is a set of states.
- \( Y \) is the set of output values.
- \( \delta_{int}: S \rightarrow S \) is the internal transition function.
- \( \delta_{ext}: Q \times X^b \rightarrow S \) is the external transition function, where
  \[ Q \in \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\} \] is the total state set,
  \( e \) is the time elapsed since last transition,
  \( X^b \) denotes the collection of bags over \( X \).
- \( \delta_{con}: S \times X^b \rightarrow S \) is the confluent transition function.
- \( \lambda: S \rightarrow Y^b \) is the output function.
- \( ta: S \rightarrow \mathbb{R}^+_0,\infty \) is the time advance function.
How an Atomic Model Works

Modeling the lecturing classroom. Output, internal event, external event?
Let’s look at an informal description of this “system”: the class lasts for 2 hours …
Internal event: class finishes
External events: fire alarm, students enter (5 minutes late, 1 hour late?)

Dr. Xiaolin Hu
Atomic Model Operation

- Ports are represented explicitly – there can be any number of input and output ports on which values can be received and sent.

- The time advance function determines the maximum lifetime in a state.

- A *bag* can contain many elements with possibly multiple occurrences of its elements. Atomic DEVS models can handle bags of inputs and outputs.

- The *external transition* function handles bags of inputs by causing an immediate state change, which also may modify the time advance.

- The *output* function can generate a bag of outputs when the time advance has expired.

- The *internal transition* function is activated immediately after the output function and causes an immediate state change, which also may modify the time advance.

- The *confluent transition* function decides the next state in cases of collision between external and internal events.
How to Define an Atomic Model

- An atomic model is a structure. To define an atomic model is to define each element of the structure, including initialization function, internal/external/confluence transition functions, output function.
- To program an atomic model in DEVSJAVA is to write java code for each of these functions.
Dr. Xiaolin Hu

Atomic Model Examples

Initialization
holdIn(“wait”, infinity)

external
If(get start message)
   If (my current state is wait)
      holdIn(“active”, 5)

start
Simple Generator

internal
If (state is active)
   holdIn(“retired”, infinity)

out
If (state is active)
   send output

pulse
time

Out
Atomic Model Examples

**Simple Generator**

- **Passive**
  - $ta = \infty$

- **Active**

**Fire-once Neuron**

- **Receptive**
  - $ta = \infty$

- **Fire**
  - Firing delay > 0
  - $ta = \infty$

- **Refract**

**Output**

- interPulseTime > 0

**External Event**

- External event

**Internal Event**

- Internal event

**Output Event**

- Output event
A DEVSESJAVA Demo of a Generator Model
(Model genr in simpArc)
Exercise: binary counter model

The system outputs a “one” for every two “one”s that it receives.
Internal Transition /Output Generation

Discussion: time advance \(=0\), \(=\) Infinity

Dr. Xiaolin Hu
Response to External Input

Make a transition

elapsed time

Time advance

Using the external transition function

Discussion:
- multiple inputs at the same time (from the same port? Different ports?)
- State transition depends on elapsed time
- Want to stay at the same state for the rest of the remaining time
- Want to generate an output immediately in response to the input, but no change on state and remaining time
- More complex: want to generate an output after a time delay in response to the input, but no change on state and remaining time (for example, the model is in the monitoring state for a pre-defined period of time. For every input it receives during the monitoring time, a corresponding output will be sent out after a delay time).
Response to Simultaneous External Input and Internal Event

Note: the output will be generated before the confluent function is executed
Discussion

1. There is no way to generate an output directly from an external input event. An output can only occur just before an internal transition.

2. To have an external event cause an output without delay, we have it “schedule” an internal state with a hold time of zero.

3. The output function does not change a model’s state.

4. In general, the only way to interact with a model is through input/output ports.

5. An implementation issue -- An atomic model works with any object-oriented classes.

6. A coupled model does not have its own states or state transition functions.
Basic Atomic Variables

**phase**

**sigma**: the scheduled remaining time in the current phase

**elapse time**: the time that elapsed in the current state
Work with Simple SISO Atomic Models
DEVSJAVA book, Chapter 2

- passive
- storage
- generator
- binaryCounter
- ramp

**Note**: the java code on the DEVSJAVA book is based on an earlier version of DEVSJAVA. They may not work for the DEVSJAVA used in this class.

Dr. Xiaolin Hu
passive model
storage model

The system responds to its input and store it forever, or until the next input comes along. Input zero signals a query. When that happens, the system sends out an output within a time, \textit{response\_time}, with the last non-zero input.
generator model

The system generates an output every time period defined by *period*. 
binary counter model

The system outputs a “one” for every two “one”s that it receives.

```java
public void initialize()
{
    count = 0;
    super.initialize();
}

public void Deltext(double e, double input)
{
    Continue(e);
    count = count + (int)input;
    if (count >= 2)
    {
        count = 0;
        holdIn("active", 10);
    }
}

public void deltint()
{
    passivate();
}

public double Out()
{
    if (phaseIs("active"))
        return 1;
    else return 0;
}
```

Dr. Xiaolin Hu
ramp model

As we shall see later, DEVS can model systems whose discrete event nature is not immediately apparent. Consider a billiard ball. As illustrated in Figure 7, struck by a cue (external event), it heads off in a direction at constant speed determined by the impulsive force imparted to it by the strike. Hitting the side of the table is considered as another input that sets the ball off going in a well-defined direction.
A switch is modeled as a DEVS with pairs of input and output ports, as shown in Figure 14. When the switch is in the standard position, jobs arriving on port “in” are sent out on port “out”, and similarly for ports “in1” and “out1”. When the switch is in its other setting, the input-to-output links are reversed, so that what goes in on port “in” exits at port “out1”, etc.

In the switch DEVS below, in addition to the standard phase and σ variables, there are state variables for storing the input port of the external event, the input value, and the polarity of the switch. In this simple model, the polarity is toggled between true and false at each input.

Code on page 50 of the DEVSJAVA book
generator that can be started/stopped and set period

pulseGenr in DEVSJAVA3_0
DEVS Coupled Model

Elements of coupled model:

- Components
- Interconnections
  - Internal Couplings
  - External Input Couplings
  - External Output Couplings
DEVS Hierarchical Modular Composition

**Atomic**: lowest level model, contains structural dynamics -- model level modularity

**Coupled**: composed of one or more atomic and/or coupled models
Coupled Model Example – Neuron net
can compute the shortest path in a directed graph by mapping distances of edges to equivalent time values.

Small network of fire-once neurons
• a pulse emitted from the generator explores two paths concurrently to reach the final neuron (number 4).
• depending on the summed firing delays along each path, a pulse emerging from one or the other will arrive first to the final neuron, the other will be shut out and prevented from continuing.
• in general, the clock time when a pulse first arrives to a neuron represents the shortest time to reach it
• if nodes and link distances are mapped to neurons and firing times, then the path taken by a pulse represents the shortest path of an associated digraph.
• the path can be reconstructed by extending the neuron model to allow retracing the path of earliest firings.

This method turns out to be isomorphic to the well known Dijkstra shortest path algorithm.

If instead of shortest paths, we request longest paths, a net of neurons that fire after their assigned fireDelays every time they receive a pulse will do the job. Finding critical paths in PERT charts require such longest path computation.
Switch Network

page 76 of the DEVSJAVA book
Generator/Processor/Transducer

![Diagram of Generator/Processor/Transducer system](image)

Page 77 of the DEVSJAVA book
Hierarchical Model

Demonstration: the GenDevsTest package
After this class

• Read Chapter 1 chapter 2, chapter 3
• Download the DEVSJAVA software from iCollege.
• Copyright form of the DEVSJAVA software.
• Set up a java project in your IDE (Eclipse is recommended) for DEVSJAVA
• **Exercise**: Copy the genr model from simpArc and put it in your package called Exercise. Modify the genr model to make it output the system clock time (by calling the `System.currentTimeMillis()`) every 20 time unit.
  • You may display the time in readable format. See https://stackoverflow.com/questions/625433/how-to-convert-milliseconds-to-x-mins-x-seconds-in-java
References:

1. B. P. Zeigler, Hessam S. Sarjoughian, Introduction to DEVS Modeling and Simulation with JAVA: Developing Component-Based Simulation Models


3. https://acims.asu.edu/