Outline

• Review of last class
• MS4 me Environment
• An carwash example – work together
• Modeling Project
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

Dr. Xiaolin Hu
Internal Transition / Output Generation

Time advance

Make a transition

Generate output

using the output function

using the internal transition function

Dr. Xiaolin Hu
Response to External Input

Make a transition using the external transition function

input

Time advance

elapsed time

Dr. Xiaolin Hu
Response to Simultaneous External Input and Internal Event

Dr. Xiaolin Hu

Make a transition

 elapsed time

Time advance

Generate output

using the confluent transition function

input

output
DEVS Coupled Model

Elements of coupled model:

• Components
• Interconnections
  – Internal Couplings
  – External Input Couplings
  – External Output Couplings
Coupling in Action

Dr. Xiaolin Hu
Basic Atomic Variables

phase

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

**elapsed time:** the time that elapsed in the current state
MS4 Me Background

• RTSync Corp. and MS4 Systems, Inc.
  – Spin off of Arizona Center for Integrative Modeling and Simulation (ACIMS) at the University of Arizona and Arizona State University
  – Founded in 2004 and 2012 respectively, as an advanced modeling and simulation consultant company
  http://acims.asu.edu
  http://www.rtsync.com
  http://www.ms4systems.com, info@ms4systems.com, support@ms4systems.com

• MS4 Modeling Environment (MS4 Me)
  – Under development for over four years
  – Commercial release, January 2013
  – Free trial version available
DEVS Modeling & Simulation Approach

- DEVS (Discrete Event System Specification) formalism provides an engine for advanced Modeling & Simulation (M&S) technology to support “build and test”.
  - DEVS evaluates different design alternative of a system, where information of system structure as well as component behavior can be implemented in an integrated simulation environment.
Structural Description: DEVS coupled model described by System Entity Structure (SES)

- The System Entity Structure (SES) is a formal framework to represent the elements of a system (or world) and their relationships in hierarchical manner.

  - **Aspects** represent ways of taking things apart into more detailed ones and labeled decomposition relation between the parent and the children.
  - **Specialization** categorizes things in specific forms and is a labeled relation that expresses alternative choices.
  - **Multi-aspects** are aspects for which the components are all of the same kind.

**Pruning:**
An operation to cut off structure in a SES that is not needed to meet particular problem specifications.
Behavior Description : DEVS Atomic Model

- DEVS model describes the behavior of a component based on state transition that displays discrete event activity of system.
Hierarchical Model Composition

- Multiple model constructs a bigger coupled model, where each model interfaced via input and output port.
Modeling and Simulation Methodology with MS4 Me

- MS4 Me provides Two Level Designs
  - High level design for domain expert
  - Low level design for modeling expert

- High level design
  - Simple Scenario
  - Sequence Diagram
  - State Diagram
  - Message types interact between models
  - Variables used in a model
  - Logics used in a model

- Low level design
  - Converting logics to computer language codes
  - Code tag block
MS4 Modeling Environment

- **Software Tool for DEVS Modeling and Simulation**
  - Based on Java and Eclipse RCP environment
- **Top down modeling methodology**
  - Sequence Diagram Designer generates a SES file and DNL files
- **Bottom up modeling methodology**
  - State Diagram Designer generates DNL files

**Features**

- Highlight keywords capability when editing SES and DNL files
- Automatic converting DNL files to Atomic Java Models
- Tree Viewer for a SES file
- Pruning process from Pruning SES document to generate coupled Java models
- Launch page to help directly access to the SES and PES files
- Support dynamic structure model
- Graphing capability
Launch Page

- New Project creates a project
- Sequence Diagram opens/creates SES files in Sequence Diagram Designer
- Atomic Model opens/creates DNL files in text editor or State Diagram Designer
- System Entity Structure (SES) opens/creates SES files in text editor
- Prune a SES Tree opens/creates PES files in text editor
- View a SES Tree displays SES file in tree view
- View Simulation opens models files and PES files in simulation viewer
Sequence Diagram and SES Document

- **Entity**
  - Represent atomic or coupled model
- **Transition**
  - Define model’s behavior
  - Message Passing

---

**From the HelloExamplesys perspective,**
**HelloExample is made of** **Person and MyWorld!**
**From the HelloExamplesys perspective,** **Person sends Hello to MyWorld!**
**From the HelloExamplesys perspective,** **MyWorld sends GoodMorning to Person!**

**From the MyWorldsys perspective,**
**MyWorld is made of** **MyRelative and MyFriend!**
**From the MyWorldsys perspective,** **MyWorld sends Hello to MyRelative!**
**From the MyWorldsys perspective,** **MyWorld sends Hello to MyFriend!**
**From the MyWorldsys perspective,** **MyRelative sends GoodMorning to MyWorld!**
**From the MyWorldsys perspective,** **MyFriend sends GoodMorning to MyWorld!**
State Diagram and DNL Document

- **Configuration**
  - Define variables and data types
  - Contain additional information

- **State**
  - Name and Time Advance
  - Initial State (Double rectangles)

- **Transition (Event)**
  - External Event (?)
  - Internal Event (!)
  - Tag blocks for Java codes

Convert State Design to DNL Document
Convert DNL Document to State Design

DNL Document:

```
accepts input on GoodMorning !
generates output on Hello !

to start, hold in sendHello for time 1!
after sendHello output Hello!
from sendHello go to waitforGoodMorning!
passivate in waitforGoodMorning !
when in waitforGoodMorning and receive GoodMorning go to passive!
passivate in passive!
```
Simulation Viewer

- Tree View of Simulation Models
- List of Variables in the selected model
- Simulation Controller
- Display Ports’ names and connections
- Simulation Status
Tutorial – a carwash example

• Chapter 4: exercise 1 part a -- car wash

• To simplify the problem, let’s first assume the system will reject any incoming cars and trucks if it is busy

• Let’s work together !!
Define the CarWashCenter Atomic model

- Right click the models.dnl to create a new atomic model.
- Launch the state designer tool. See figure.
- Design the states and state transitions
  - We first assume the system rejects any incoming cars/trucks if busy.
  - We assume the processing time is 10.
- Always save before exist.
- Test the atomic model by clicking “View in Simviewer”
- Add a variable processingTime using the Configuration tool in the state designer window.
- Use the variable processingTime
  - A variable needs to be used in quotation, e.g., “processingTime”
- See the DNL file.
- See the JAVA file.
  - Do not change using the Java file directly because it will cause inconsistency between the java source code and the DNL (and the state designer).

Dr. Xiaolin Hu
DNL file

```dnl
use processingTime with type double and default "10"!
accepts input on car!
generates output on job!

to start passivate in passive!
when in passive and receive car go to active!
hold in active for time "processingTime"
after active output job!
from active go to passive!
```

You can define models using DNL directly (without using the state designer tool).

Dr. Xiaolin Hu
Test the model by coupling with a CarGenerator

- Create a carGenerator
  - Assuming the inter generation time is 15
- Test the carGenerator model
- Coupled the two models
  - Right click the Models.ses to create a new System Entity Structure (coupled model)
  - Launch the Sequence Designer Tool for the created system entity structure
  - Design the couplings between the two models.
  - Note: it is important for the entity names to match the atomic model names, and for
    the names to match the output port and input port names of the source and
    destination models. By default, the input and/or output port names are the same as
    the message name specified in both the state designer tool and the sequence
    designer tool.
- See the generated DNL file
- Test the model by clicking the “Simulate Now”
- Add an external output coupling so we can see the output from the
carWashCenter
  - In sequence designer, add an entity and select “Top Level”
  - Add an coupling from the carWashCenter to the top level model
Tutorial (cont.) – A more complex generator

• Add random number in carGenerator and generate cars with a sequence number
  – Import java.util.Random
  – Add an variable ran and initialize it by creating an instance of Random object
  – Add a variable interGeneTime
  – Note: you can do the above three things through the Configuration button in the State Designer tool (see next slide as an example), or write code (either DNL code or embedded java code) directly in the DNL file. See the final DNL file as an example.
  – Use the internal event tagging to set interGeneTime as a random number (An example of launching the tagging tool for external event transition is given later).
  – Change the active period to be “interGeneTime”.
  – Test the model.
  – Add a sequence variable with default value to be 0
  – Launching the output event tagging tool to send the output string with the sequence number.
  – Test the model.
Configuration for More information Example
Example of Launching the Tagging tool
The carGenerator DNL File

```dnl
use ran with type Random and default "null"!
use interGeneTime with type double and default "15"!
use sequence with type int and default "0"!

generates output on car!

Initialize variables
<%ran = new Random(123);%>!

to start hold in active for time "interGeneTime"!
after active output car!
from active go to active!

output event for active
<%output.add(outcar,"car"+sequence);%>!

internal event for active
<%interGeneTime = ran.nextInt(20);
sequence++;%>!

add library
<%import java.util.Random;
%>
```

Dr. Xiaolin Hu
Tutorial (cont.) – Add complexity

• Make the carWashCenter remember the incoming job.
  – Add a String variable currentJob
  – Open the external event tagging tool to set the value of the currentJob
  – Send the currentJob out as the output message

• Test the coupled model in SimView

• Create a truck generator (copy the carGenerator then modify)

• Modify the carWashCenter to handle truck input
  – Note: the processingTime is different

• Add the truck generator to the system using the sequence designer

• Add queue to the carWashCenter

• Display the queue size in simulation. -- getTooltip;

• Add a transducer
  – Use the keyword “eventually” if an external event does not need to change the existing schedule.

• Each car/truck has its own processing time and other properties such as price and priority. --- add a vehicleEntity class. – Not covered. Talk to me if you are interested.

Dr. Xiaolin Hu
Exercise

• The car wash center can have at most 5 cars waiting in the queue.
  – Check the size of the queue
  – Use different phases, e.g., first, second…..

• The car wash time depends on the number of cars in the queue. For example, the car wash time $t = T \times (1/2)^n$, where $n$ is the number of cars in the queue.

• There are two service lanes.
  – Atomic model
  – Coupled model, similar to the simpArc example
Homework

- Download the DEVSTutorialLab from the class webpage
  - Create a new project
  - Copy the four .dnl files to the \src\Models\dnl folder
  - Copy the .ses file to the \src\Models\ses folder
  - Refresh your project in MS4Me, the java files will atomically generated.

- Next class I’ll have 15min Q&A section for MS4Me at the end of the class.

- Description of the hands-on project is posted on class webpage

- Form the project group and decide your project topic. Then email your group and project information to me by next week (March 11).
  - Type
  - A brief description of the project
Project Topic Ideas

• **Workflow modeling** (process + time-based reminder + simulation-based evaluation of potentially large number of cases)
  
  – Business process workflow
    • Example: from orders placed online, to ordered shipped from warehouse and eventually delivered to the customers.
  
  – Medical pathways
  
  – Hospital process workflow
    • Example: medical billing process, patient admission and discharge process.