The Relationship Between DEVS Models and Real Systems

2024-SIW-Presentation-021

Rob Kewley, simlytics.cloud, USA
Bernard Zeigler and Doohwan (DH) Kim, RTSync Corporation, USA
Key Takeaway

• Science of systems engineering manages complexity by breaking a large system into sub-systems/components and dealing with interfaces
  ▪ Enables reasoning at different levels of complexity
  ▪ Enables modularity

• DEVS allows us to manage the complexity of our system models the same way
Agenda

• **System models**
• **Discrete Event Specification System (DEVS) Models**
  ▪ DEVS Atomic Models
  ▪ DEVS Coupled Models
  ▪ Hierarchies of Coupled Models
• **Joint Enterprise Example**
  ▪ System Model in SysML
  ▪ System Entity Structure
  ▪ DEVS annotations in SysML
  ▪ Generating DEVS Models from SysML
• **Work to do**
• **Conclusions**
• System is a black box
• Stream of inputs produces stream of outputs
If we know the current state, and the stream of inputs at a specific time, the state transition function determines the next state.

The output function determines the output from the current state.

Mathematically based on Moore finite state machine.
In a coupled model, individual models are connected to each other and to external inputs and outputs.

Closure under coupling – when viewed as a black box, the coupled model behaves the same as the system model.

Formulation as a Moore machine eliminates zero-delay feedback loops.
In a coupled model, individual models are connected to each other and to external inputs and outputs.

Closure under coupling – when viewed as a black box, the coupled model behaves the same as the system model.
Hierarchy of Models

- The closure under coupling property enables the construction of a hierarchical network of models.
- This idea is the foundation of modular model composition.
Hierarchy of Models

- The closure under coupling property enables the construction of a hierarchical network of models.
- This idea is the foundation of modular model composition.
- Hierarchies can be of any depth and complexity.
Parallel DEVS Model

- **Parallel discrete event case of system model**
- **Interacts with external systems via messages only (Leads to distributed, loosely coupled models)**
- **Adds time advance function to schedule next internal event for a model**
- **Breaks state transition function into 3 cases**
  - Internal state transition executes next internal event
  - External state transition based on receipt of inputs prior to next internal event
  - Confluent transition function executed when inputs are received as same time as next internal event
Coupled Parallel DEVS Model

- Interface is the same as a Parallel DEVS Model
- Routes input/output streams to the influenced models
- Manages time and execution of subordinate models
- Yields discrete event model that allows stream of inputs, outputs and state to be computed for models in the structure
What about MBSE? Wymore’s T3SD/DEVS

- Real systems are technology components, connected by couplings, that perform desired functions to transform inputs to outputs
- DEVS allows the hierarchical simulation of different designs at different levels of abstraction
- DEVS allows a mapping of hierarchical system components to hierarchical system models that supports modularity

Joint Enterprise SysML Block Definition Diagram
Joint Enterprise System Behavior

store : Store

plant : Plant

1: order(Order)

1.1: Request build(Assembly Line, Build Request)
1.2: Build Product
1.3: Ship(Shipment)

2: Deliver(Shipment)

shipper : Shipper
Joint Enterprise System Entity Structure

Joint Enterprise

Stores
- Store 1: Holiday Demand, Normal Demand
- Store 2: Holiday Demand, Normal Demand

Order Process

Build Request Process

Plant

Assembly
- Current Assembly
  - Assembly Line 1: Product A Rate
  - Assembly Line 2: Product B Rate
- Future Assembly
  - Assembly Line 3: Product A Rate
  - Assembly 3: Product B Rate

Shipping
- Van Capacity
- Truck Capacity

Vehicle
Retailer

- **Store Model**
  - Internal state
    - Address
    - Product Demand
    - Product Stock
- **Ports**
  - Customer arrival
  - Customer departure
  - Receive shipment
  - Send order
- **Operations**
  - Process customer
  - Ordering
  - Process delivery
- **Data objects**
  - Product Demand
  - Product Stock
  - Customer
  - Order (not shown)
Retailer Model Definition
Store Experimental Frame Internal Block Definition
Store Model Internal Block Definition
SysML Model Produces XMI Data

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="20131001">...
</xmi:XMI>
</?xml version="1.0" encoding="UTF-8"?>
```
Generate DEVS Model from System

System Parser
• Scala program
• Parses XML file representing system
• Builds data structures representing DEVS models
• Passes data to generator
• Could write other parsers for different sources of system data:
  ▪ Database
  ▪ SysML V2 API
  ▪ Other modeling tool

Code Generator
• Scala program
• Receives data structures representing DEVS models from parser
• Generates code structure for
  ▪ Supporting data types (Immutable Framework)
  ▪ DEVS Models (DEVS Streaming Framework)
  ▪ Coupled Models (DEVS Streaming Framework)
  ▪ Couplings
  ▪ Simulators to run the models
• Developer must specify behavior in the transition functions, time advance function, and output function
• Could write other generators for other types of DEVS models
  ▪ adevs
package iso.example.store;
import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import devs.Port;
import devs.msg.Bag;
import devs.msg.PortValue;
import devs.msg.time.LongSimTime;
import devs.PDEVSModel;
import java.util.List;
import iso.example.store.*;

public abstract class StoreModel extends PDevsModel<LongSimTime, StoreModel.StoreModelState> {
    public static Port<Customer> customerArrival = new Port<>(CUSTOMER_ARRIVAL);
    public static Port<Customer> customerDeparture = new Port<>(CUSTOMER_DEPARTURE);
    public static Port<Shipment> receiveShipment = new Port<>(RECEIVE_SHIPMENT);
    public static Port<Order> sendOrder = new Port<>(SEND_ORDER);

    public static class StoreModelState {
        protected String address;
        protected List<ProductDemand> productdemand;
        protected List<ProductStock> productstock;
        public StoreModelState(String address, List<ProductDemand> productdemand, List<ProductStock> productstock) {
            this.address = address;
            this.productdemand = productdemand;
            this.productstock = productstock;
        }
    }

    public StoreModel(StoreModelState initialState, String identifier) {
        super(initialState, identifier);
    }

    private abstract void processCustomer(Customer customer);
    private abstract void ordering(Order order);
    private abstract void processDelivery(Shipment shipment);
}
Work to do...

• Develop code generator for adevs model
• Develop code generator for distributed models using DEVS Streaming Framework and Apache Kafka streams
• Iterate with other examples
• Investigate other code generation methodologies
Conclusion

• DEVS models are derived from the mathematics of system models
• DEVS models are structurally and semantically similar to system models
• DEVS models can be derived from SysML representations of real systems
  ▪ Requires annotating with model-specific relationships
  ▪ Generate DEVS code from XMI produced by SysML model

• Science of systems engineering manages complexity by breaking a large system into sub-systems/components and dealing with interfaces
  ▪ Enables reasoning at different levels of complexity
  ▪ Enables modularity
• DEVS allows us to manage the complexity of our system models the same way
Q&A / Discussion