Introducing Messages in Modelica for Facilitating Discrete-Event System Modeling

Victorino Sanz, Alfonso Urquía and Sebastián Dormido
Dpto. Informática y Automática, E.T.S.I. Informática, UNED
Juan del Rosal 16, 28040, Madrid, Spain
{vsanz,aurquia,sdormido}@dia.uned.es
Outline

- Introduction
- Process-Oriented Modeling
  - ARENAlib
  - SIMANLib
- Parallel DEVS
  - DEVSLib
- Messages in Modelica
- Conclusions

Victorino Sanz, Alfonso Urquia and Sebastián Dormido
Dpto. Informática y Automática, E.T.S.I. Informática, UNED.
{vsanz,aurquia,sdormido}@dia.uned.es
Introduction

- Objective: process-oriented modeling using the Modelica language
- Modelica capacities for discrete-event system modeling
  - *If*-expressions
  - *when*-clauses
- Previous works use event-scheduling
  - State Machines, Petri Nets, DEVS,...
New Modelica libraries have been developed for process-oriented modeling

- ARENALib, SIMANLib

Parallel DEVS formalism has also been considered

- DEVSLib

Complex solutions to problems and still some restrictions

Introduction of the messages mechanism for facilitating discrete-event system modeling

- Proposed implementation
Process-Oriented Modeling

- System observed from the point of view of an entity
- Entities flow through the system and are processed using resources. If no resources are available, entities usually wait in queues
- Flow of information between components
- Examples:
  - Bottle manufacturing process
  - Airport check-in system
  - Bank teller
Process-Oriented Modeling

- ARENALib
  - Based on the Arena simulation environment
  - Components: flowchart and data modules
  - Entities are transferred between flowchart modules

Victorino Sanz, Alfonso Urquía and Sebastián Dormido
Dpto. Informática y Automática, E.T.S.I. Informática, UNED.
{vsanz,aurquia,sdormido}@dia.uned.es
Process-Oriented Modeling

- Model Communication
  - Direct Transmission
    - Entities defined with variables inside the connector
    - Problem with the simultaneous reception of entities
      - Semaphores: poor performance
      - Flow variable: connector cannot contain several entities at the same time
  - Text File approach
    - Text file as intermediate storage for entities
    - Poor performance and low flexibility.

Victorino Sanz, Alfonso Urquía and Sebastián Dormido
Dpto. Informática y Automática, E.T.S.I. Informática, UNED.
{vsanz,aurquia,sdormido}@dia.uned.es
Process-Oriented Modeling

- Model Communication (II)
  - Dynamic Memory Storage
    - Substitute the text file with dynamic memory space
    - Increases performance
    - Higher flexibility due to independency between data structures and their management
  - Entity Management
    - Temporal storage for delayed entities.
    - Use of dynamic memory storage.

Victorino Sanz, Alfonso Urquia and Sebastián Dormido
Dpto. Informática y Automática, E.T.S.I. Informatica, UNED.
{vsanz,aurquia,sdormido}@dia.uned.es
Process-Oriented Modeling

- Stochastic Data Generation
  - CMRG and RandomLib
- Statistical Information Management
  - Statistical indicators.

<table>
<thead>
<tr>
<th>Module</th>
<th>Indicator</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>System.NumberIn</td>
<td>Obs</td>
</tr>
<tr>
<td>Process</td>
<td>NumberIn</td>
<td>Obs</td>
</tr>
<tr>
<td></td>
<td>NumberOfOut</td>
<td>Obs</td>
</tr>
<tr>
<td></td>
<td>VATime Per Entity</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td></td>
<td>NVATime Per Entity</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td></td>
<td>TotalTime Per Entity</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td></td>
<td>Queue.NQ</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td></td>
<td>Queue.WaitTime</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td>Dispose</td>
<td>System.NumberOut</td>
<td>Obs</td>
</tr>
<tr>
<td>EntityType</td>
<td>NumberIn</td>
<td>Obs</td>
</tr>
<tr>
<td></td>
<td>NumberOfOut</td>
<td>Obs</td>
</tr>
<tr>
<td></td>
<td>VATime</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td></td>
<td>NVATime</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td></td>
<td>TranTime</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td></td>
<td>WaitTime</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td></td>
<td>OtherTime</td>
<td>Avg, Min, Max, Final, Obs</td>
</tr>
<tr>
<td></td>
<td>Work In Progress</td>
<td>Avg, Min, Max, Final</td>
</tr>
</tbody>
</table>
Process-Oriented Modeling

- **SIMANLib**
  - Reproduces some basic functionalities of the SIMAN simulation language
  - Components: blocks and elements
Parallel DEVS

- **DEVSLib**
  - Library Architecture
  - Model Development
    - Atomic model
      1. Duplicate atomicDraft
      2. Include input and output ports
      3. Redeclare fext, fint, fout, ta, initst and st
    - Coupled model
      1. Include input and output ports
      2. Include components (atomic or coupled models)
      3. Connect components

```java
model processor
extends AtomicDEVS(
  redeclare record State = st);
redeclare function Fcon = con;
redeclare function Fint = int;
redeclare function Fext = ext;
redeclare function Fta = ta;
redeclare function initState = initst(dt=processTime);
parameter Real processTime = 1;
Interfaces.outPortManager
outPortManager1(n=1,
  redeclare record State = st,
  redeclare function Fout = out);
Interfaces.outPort outPort1; // output port
Interfaces.inPort inPort1; // input port
equation
iEvent[1] = inPort1.event;
iQueue[1] = inPort1.queue;
connect(outPortManager1.port,outPort1);
end processor;
```

Victorino Sanz, Alfonso Urquía and Sebastián Dormido
Dpto. Informática y Automática, E.T.S.I. Informatica, UNED.
{vsanz,aurquia,sdormido}@dia.uned.es
Parallel DEVS

- Modeling Restrictions
  - One-to-Many connections. DUP model.

- Static information structure transmitted at events
  - Current: type and value
  - Additional types: arrays, records,...
Messages in Modelica

- Simple mechanism for communicating information between models
- Components: message and mailbox
- Characteristics
  - Messages can be sent to any available mailbox
  - Mailboxes warn of new incoming messages
  - Messages are read from the mailbox
  - Transmission of messages is instantaneous.
  - Content of the message independent from its type
  - Messages can be received simultaneously
  - Two stages in the communication: send and reception

Victorino Sanz, Alfonso Urquia and Sebastián Dormido
Dpto. Informática y Automática, E.T.S.I. Informatica, UNED.
{vsanz,aurquia,sdormido}@dia.uned.es
Messages in Modelica

- Communication using messages

- Mailboxes can be shared between models

- Mailboxes can be included in connectors

- The user has to define the treatment of each message

Victorino Sanz, Alfonso Urquia and Sebastián Dormido
Dpto. Informática y Automática, E.T.S.I. Informática, UNED.
{vsanz,aurquia,sdormido}@dia.uned.es
Messages in Modelica

- Proposed Implementation
  - Data Structures
    - Message: type and reference to the content
    - Mailbox: reference to temporary storage space
  - Operations.
    - Mailbox operations
      - Newmailbox, checkmsg, newmsg, nummsg, readmsg, getmsg, putmsg
    - Message operations
      - Newmsg, gettype, settype, getcontent, setcontent
Conclusions

- Process-oriented modeling with Modelica is a difficult task
- Several Modelica libraries have been developed for process-oriented modeling.
- Its implementation present problems and restrictions, and the applied solutions are complex
- Messages mechanism facilitates the transmission of information between models
- Messages simplify the development of discrete-event system models