Towards a Formal Standard for Interoperability in M&S/System of Systems Integration

Bernard Zeigler, Saurabh Mittal
Arizona Center for Integrative Modeling and Simulation,
University of Arizona,
Tucson, AZ
(zeigler | saurabh)
@ece.arizona.edu

Xiaolin Hu
Dept of Computer Science,
Georgia State University,
Atlanta, GA
xhu@cs.gsu.edu

Outline

• Systems interoperability (vs integration)
• Roles of Modeling and Simulation in System of Systems
• Why middleware (HLA) is not enough
• Levels of Interoperability – from conceptual to linguistic
• Testing interoperability at multiple levels
• DEVS standard for simulation interoperation
• Application to testing the GIG/SOA
• Summary
Interoperation vs Integration*

Interoperation of components
- participants remain autonomous and independent
- loosely coupled
- interaction rules are soft coded
- local data vocabularies persist
- share information via mediation

Integration of components
- participants are assimilated into whole, losing autonomy and independence
- tightly coupled
- interaction rules are hard coded
- global data vocabulary adopted
- share information conforming to strict standards


Problem formulation: Systems of Systems

System of Systems (SoS)
- interoperate disparate systems to synthesize a new functionality
- defining obstacle is lack of interoperability among components

M&S as Smart Component

M&S as Solution Methodology
Tolkien’s Levels of Conceptual Interoperability Model

<table>
<thead>
<tr>
<th>Level of Conceptual Interoperability</th>
<th>Characteristic</th>
<th>Key Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual</td>
<td>The assumptions and constraints underlying the meaningful abstraction of reality are aligned</td>
<td>Requires that conceptual models be documented based on engineering methods enabling their interpretation and evaluation by other engineers.</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Participants are able to comprehend changes in system state and assumptions and constraints that each is making over time, and are able to take advantage of those changes.</td>
<td>Requires common understanding of system dynamics.</td>
</tr>
<tr>
<td>Pragmatic</td>
<td>Participants are aware of the methods and procedures that each is employing.</td>
<td>Requires that the use of the data – or the context of their application – is understood by the participating systems.</td>
</tr>
<tr>
<td>Semantic</td>
<td>The meaning of the data is shared.</td>
<td>Requires a common information exchange reference model.</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Introduces a common structure to exchange information.</td>
<td>Requires that a common data format is used.</td>
</tr>
<tr>
<td>Technical</td>
<td>Data can be exchanged between participants.</td>
<td>Requires that a communication protocol exists.</td>
</tr>
<tr>
<td>Stand-alone</td>
<td>No interoperability.</td>
<td></td>
</tr>
</tbody>
</table>

Linguistic Levels of Interoperability

<table>
<thead>
<tr>
<th>Linguistic Level</th>
<th>Interoperability Demonstrated if:</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pragmatic –</td>
<td>The receiver reacts to the message in a manner that the sender intends.</td>
<td>A commander’s order is obeyed by the troops in the field as the commander intended. (This assumes semantic interoperability.)</td>
</tr>
<tr>
<td>How information in message is used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic –</td>
<td>The receiver assigns the same meaning as the sender did to the message.</td>
<td>An order from a commander to multinational participants in a coalition operation is understood in the same manner despite translation into different languages.</td>
</tr>
<tr>
<td>Shared understanding of meaning of messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntactic –</td>
<td>The consumer is able to receive and parse the sender’s message.</td>
<td>A common network protocol (e.g., IPv4) ensures that all nodes on the network can send and receive data bit arrays while adhering to a prescribed format.</td>
</tr>
<tr>
<td>Common rules governing composition and transmitting of messages</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Mapping M&S Layers to Linguistic Levels

**Syntactic Level**
- Abstract Simulators, Real time Execution, Animation Visualization

**Semantic Level**
- Distributed Grids, Service Oriented Architectures
- Ontologies, Formalisms, Model Dynamic Structure, Life Cycle Continuity, Model Abstraction

**Pragmatic Level**
- SES, DoDAF, Integrated System Development and Testing

**Execution Layer**
- Design and Test Development Layer
- Observers and Agents for Net-Centric Key Performance Parameters

**Network Layer**
- Collaboration Layer
- Semantic Web, Composition, Orchestration

**Background: DEVS M&S Framework**

**Discrete Event Systems Specification (DEVS)**
- Based on mathematical formalism using system theoretic principles
- Separation of Model, Simulator and Experimental Frame
- Atomic and Coupled types
- Hierarchical modular composition

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>System Specification at this level</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Coupled Systems</td>
<td>System built from component systems with coupling recipe.</td>
</tr>
<tr>
<td>3</td>
<td>IO System Structure</td>
<td>System with state and state transitions to generate the behavior.</td>
</tr>
<tr>
<td>2</td>
<td>IO Function</td>
<td>Collection of input/output pairs constituting the allowed behavior partitioned according to initial state of the system. The collection of I/O functions is infinite in principle because typically, there are numerous states to start from and the inputs can be extended indefinitely.</td>
</tr>
<tr>
<td>1</td>
<td>IO Behavior</td>
<td>Collection of input/output pairs constituting the allowed behavior of the system from an external Black Box view.</td>
</tr>
<tr>
<td>0</td>
<td>IO Frame</td>
<td>Input and output variables and ports together with allowed values.</td>
</tr>
</tbody>
</table>
DEVS Modeling and Simulation Infrastructure supports simultaneous testing at multiple levels

<table>
<thead>
<tr>
<th>DEVS Modeling Language (DEVML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVS Simulator Services</td>
</tr>
<tr>
<td>Middleware (SOAP, RMI etc)</td>
</tr>
<tr>
<td>Net-centric infrastructure</td>
</tr>
</tbody>
</table>

DEVS Simulation Concept

- Specifies the abstract simulation engine that correctly simulates DEVS atomic and coupled models
- Gives rise to a general protocol that has specific mechanisms for:
  - declaring who takes part in the simulation:
    - format for referencing federates (participants)
  - declaring how federates exchange information:
    - format for their message exchange patterns
  - executing an iterative cycle that
    - controls how time advances:
      - updating the clock based on next event times
    - determines when federates exchange messages:
      - the point in the cycle when all interchange takes place
    - determines when federates do internal state updating:
      - the point in the cycle when next event times are collected

**Note:**
If the federates are DEVS compliant then the simulation is **provably correct** in the sense that the DEVS closure under coupling theorem guarantees a well-defined resulting structure and behavior.
DEVS Simulation Protocol

Concept of DEVS Standard
Core Simulator Interface

```java
interface coreSimulatorInterface{
    void setSimulators(Collection<CoreSimulatorInterface>);
    void initialize();
    Double nextTN();
    void computeInputOutput(Double t);
    void applyDeltFunc(Double t);
    void putContentOnSimulator(CoreSimulatorInterface sim, ContentInterface c);
    void sendMessages();
}
```

Core Simulator Interface is derived from the DEVS simulation cycle
It specifies the methods and arguments to be coordinated under the DEVS protocol

DEVS/SOA Infrastructure: Supports Deployment and Execution of DEVS Models on the Web

- Service Oriented Architecture (SOA) consists of various W3C standards
- Client server framework
- XML Message encapsulated in SOAP wrapper
- Machine-to-machine interoperability over the network based on WSDL interface descriptions
DEVS/SOA Infrastructure for GIG Mission Thread Testing

1. MAJ Smith tasks Intell to reconnoiter objective area and provide threat estimate.

2. Posts tasking using Discovery and Storage.

3. Intell Cell initiates high priority collection against objective, and collectors post raw output.

4. Intelli posts products via Discovery and Storage.

5. MAJ Smith pulls estimate from Storage

6. MAJ Smith computes time for task, measures performance.

Observing Agent for Major Smith
Observing Agent for Intell Cell

Computes Time for Task, Measure Performance

Summary

The proposed DEVS standard and its DEVS/SOA implementation support several modes:

DEVS-to-DEVS Interoperability
- DEVS standard facilitates interoperability at the syntactic, semantic and pragmatic levels.

DEVS-to-Non-DEVS Interoperability
- Direct
  - Refactoring legacy simulations to implement the Core Simulator interface.
  - Allows interoperation with DEVS and other non-DEVS peers.
  - Guarantees well-defined time management and simulation correctness.
  - Sound basis for interoperability at the higher levels.
- Via Client Gateways
  - SOA standard enables interoperation of services (DEVS and non-DEVS).
  - DEVS/SOA can deploy DEVS models to act as agents that are automatically attached to clients.
  - Test agents can
    - Observe the web service interactions between client and server.
    - Serve as virtual users to interact with other users.
    - Direct the course of test scenarios.
    - Communicate with each other to coordinate and share information.
Backup

Layered structure

- Atomic and Coupled Simulators Interfaces
- DEVS Modeling Interfaces
- DEVS Simulator Interfaces
- DEVS Supporting Interface
- Atomic and Coupled Model Interfaces
- Entity, and Collection, Message Interfaces
DEVS Supporting Interfaces

interface EntityInterface{
    String getName();
    boolean equalName(String name);
}

interface Collection extends EntityInterface{
    int size();
    void add(EntityInterface entity);
    void remove(EntityInterface entity);
    boolean contains(EntityInterface entity);
}

Message-related interfaces

interface ContentInterface {
    PortInterface getPort();
    EntityInterface getValue();
    boolean onPort(PortInterface port);
}

interface PortInterface extends EntityInterface{
}

interface MessageInterface extends Collection{
    boolean onPort(PortInterface port, ContentInterface content);
    EntityInterface getValOnPort(PortInterface port, ContentInterface content);
}
Ensemble Interfaces

```
interface ensembleBasic {
    void tellAll(Method m, EntityInterface[] args);
    ensembleCollection askAll(Method m);
    ensembleCollection which(Method m);
    EntityInterface whichOne(Method m);
}

interface ensembleCollection extends ensembleBasic, Collection{
    public ensembleCollection copy(ensembleCollection ce);
}
```

DEVS Model Interfaces

```
interface IODevs {
    void addInport(String portName);
    void addOutport(String portName);
    ensembleCollection getInports();
    ContentInterface makeContent(PortInterface port, EntityInterface value);
    boolean messageOnPort(MessageInterface x, PortInterface port, ContentInterface c);
}

interface basicDevs {
    void deltext(double e, MessageInterface x);
    void deltcon(double e, MessageInterface x);
    void deltint();
    MessageInterface Out();
    double ta();
    void initialize();
}

interface coupledDevs {
    void add(IODevs d);
    void addCoupling(IODevs src, Port p1, IODevs dest, Port p2);
    IODevs getComponentWithName(String nm);
    ensembleCollection getComponents();
    ensembleCollection getCouplings(IODevs src, Port p1);
}

interface IOBasicDevs {
    void addInport(String portName);
    void addOutport(String portName);
    ensembleCollection getInports();
    ensembleCollection getOutports();
    ContentInterface makeContent(PortInterface port, EntityInterface value);
    boolean messageOnPort(MessageInterface x, PortInterface port, ContentInterface c);
}
```
DEVS Simulator Interfaces

See also

A Proposed DEVS Standard: Model and Simulator Interfaces, Simulator Protocol
Xiaolin Hu
Bernard P. Zeigler

On

http://osa.inria.fr/wiki/NCMS/NCMS
Books and Web Links