Towards a Theory for Systems of (IT) Systems Engineering: Interoperability and Reusability in the DEVS Framework

Bernard P. Zeigler
Arizona Center for Integrative Modeling and Simulation
C4I Center, GMU
and
RTSync Corporation
Outline

• Faster, cheaper computers, networking, and web middleware are explosively driving up the complexity of Information Technology systems.

• Model-driven methodology is being applied to IT system development, but does not address crucial issues in system integration and interoperability.

• With this motivation, we work toward a theory of System of Systems (SoS) to clarify issues in interoperability, integration, composability, and reuse.

• Finally, we indicate how these concepts are made operational by the Discrete Event Systems Specification (DEVS) framework and the Global Information Grid/Service Oriented Architecture environment.
Issues In Developmental Complexity of IT Systems

• Often development does not start from scratch
• Conditioned by idiosyncratic requirements
• Powered, but unconstrained, by applicable standards
• Requires legacy subsystem “integration”
• Rigorous testing is needed to cope with complexity
• Methodology must scale with growth and evolution of system

UML/MDA offers only limited support to address these concerns
Global Information Grid/Service Oriented Architecture

source: http://data2use.com/share/DISA_CustomerPartnershipConf-NCES_Briefs.zip
Formulate the Issues within a System of IT Systems Conceptual Framework

- SoS = collection of disparate IT systems to be federated to satisfy new requirements
- Each participating system may itself be large and complex
- Existing systems usually have become efficient at achieving their own specialized requirements
- Existing systems often adhere to idiosyncratic formalisms and development approached
- How do we make existing systems work together? How do we design new systems for easier federation?

SoS Engineering (SOSE) is the discipline that engineers solutions to SoS problems

M&S, DEVS and systems theory potentially offer a complete solution to the SOSE problem
Some Ancient System Theory

SoS – Definitions & Perspectives

System – minimal properties:
• has an identity - can be distinguished from other things
• is capable of federation –
  – can be a federate and
  – can be federated with other federates
  – SoS: is a System resulting from federation of Systems

Lifecycle Dynamic Structure:
• Free Standing
• Federated in one or more SoS
  – Interoperating with others – retains identity
  – Integrated with others to form new Free Standing System – loses identity
  – Continuum between two
Towards a Theory of (IT) SoSE

2\textsuperscript{nd} level SoS

1\textsuperscript{st} level Systems

Primitives

Reusability Framework

Composition Framework

Support for achieving correct interaction at the syntactic, semantic, and pragmatic levels

New requirements formulated as pragmatic frame

Support for creating Federations – e.g., HLA or DEVS coupled model

SoS

Interoperability/Integration Support

Federation Support

Systems
Reusability Framework

• Synthesized first level systems have large invested, learned knowledge content making reusability desirable
• But constraints emerge due to dynamics of, and content embedded in, systems
• Federation Support – middleware to enable information exchange
• interoperation/integration Support –
  – content compatibility – introduce linguistic levels
  – reuse effectiveness – activity-based correlation
• For scalability, need to support hierarchical composition
Review: DEVS Framework

Discrete Event Systems Specification (DEVS) is the basis for a formal framework for modeling and simulation:

- Exploits the separation between model, experimental frame and simulator
- Offers a standard for distributed simulation to support interoperability, composability, and reuse
- Supports automated, integrated complex systems development and testing
- Provides infrastructure for rigorous distributed simulation-based testing
- Supports Systems Entity Structure/Pragmatic Frame ontology development
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>I/O System Structure</td>
<td>System with state and state transitions to generate the behavior.</td>
</tr>
<tr>
<td>2</td>
<td>I/O 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>I/O 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>I/O Frame</td>
<td>Input and output variables and ports together with allowed values.</td>
</tr>
</tbody>
</table>
Some Types of Models Represented in DEVS

Atomic Models
- Ordinary Differential Equation Models
- Spiking Neuron Models
- Petri Net Models
- Discrete Time/StateChart Models
- Quantized Integrator Models
- Fuzzy Logic Models
- Reactive Agent Models

Coupled Models
- Processing/Queuing/Coordinating Networks
- Networks, Collaborations
- Processing Networks
- Spiking Neuron Networks
- Multi Agent Systems
- Self Organized Criticality Models
- Partial Differential Equations
- Physical Space
- n-Dim Cell Space
- Cellular Automata

can be components in a coupled model
DEVS Simulation Concept

DEVS Model Specification

DEVS Simulation Protocol

DEVS Model

DEVS Protocol

DEVS Simulator

DEVS Protocol 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
  • declaring how federates exchange information
  • executing an iterative cycle that
    ✓ controls how time advances
    ✓ determines when federates exchange messages
    ✓ determines when federates do internal state updating

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.
Interoperation vs Integration*

Interoperation of systems
• participants remain autonomous and independent
• loosely coupled
• interaction rules are soft coded and encapsulated
• local data vocabularies and ontologies for interpretation persist
• share information via mediation
• asynchronous data transfer

Integration of systems
• participants are assimilated into whole, losing autonomy and independence
• tightly coupled
• interaction rules are hard coded and co-dependent
• global data vocabulary and ontology for interpretation adopted
• share information conforming to strict standards
• synchronous data transfer


NOT Polar Opposites!

SPECTRUM of INTERACTION MODES

reusability
composability
flexibility

fit-to-purpose
responsiveness

Example:
Kill chain combines interoperation and integration modes

• The kill chain illustrates the co-existence of interoperation and integration modes of component interaction.
• Early activities in the chain are characterized by larger field of view and have more information-centric functions than do later activities. They need the loose coupling and flexibility of interoperation.
• Later activities are more action-centric requiring the tight coupling and responsiveness of integrated components.
### Linguistic Levels of Information Exchange and Interoperability

<table>
<thead>
<tr>
<th>Linguistic Level of Information Exchange</th>
<th>A System of Systems interoperates at this level if:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pragmatic</strong> – how information in messages is used</td>
<td>The receiver re-acts to the message in a manner that the sender intends (assuming non-hostility in the collaboration).</td>
</tr>
<tr>
<td><strong>Semantic</strong> – shared understanding of meaning of messages</td>
<td>The receiver assigns the same meaning as the sender did to the message.</td>
</tr>
<tr>
<td><strong>Syntactic</strong> – common rules governing composition and transmitting of messages</td>
<td>The consumer is able to receive and parse the sender’s message</td>
</tr>
</tbody>
</table>

**System Participant**

- **pragmatic**
- **semantic**
- **syntactic**

**System Participant**

- **pragmatic**
- **semantic**
- **syntactic**
Pragmatic level agents collaborate to guide and observe mission thread executions.

Semantic level agents observe pair-wise information exchanges and check for meaning and timing.

Syntactic level agents return statistics and alarms to higher layer level agents.

Net-Centric Test Agent Capability (NTAC)
Survival of Systems: Deriving a principle for selection of components

System of Component Systems

Environment

Energy earned from the environment due to action

Energy expended in order to take the action

Survive if energy acquired >= energy expended

Select components based on their contribution to energy cost/benefit ratio:
How can we formulate this in an appropriate abstraction?
Activity— the mediating concept

Activity relates information processing work to energy consumed -- activity distribution among components is represents an abstraction of energy
Activity-Aware System Architecture

System

performer

Situation characterization

Decision Making

Action

Activity Measurement

Input/output Evaluation

Structure Search And Change

Feedforward – what is the problem? How have we solved it in the past?

Internal Feedback – how much energy did it cost?

• Persistent record of component achievements
• Reuse to populate initial search
• Update after search

External Feedback – how did we do? (energy acquired)

Decomposed Internal Feedback – how much did each component contribute? (credit assignment)
Component Rating = Activity*Outcome
Correlation Over Trials

Credit assigned = activity * outcome score
Component rating = accumulation of credit assigned over trials
Search supported by Pruning biased by Achievement levels

Bias search with past achievement levels after a start up phase.

Automated pruning

Check if this composition already tried

Score activityResult components

Composition Log

Achievement Log
DEVS Agent Supported Credit-Enhanced Component Re-use
Summary

• Model-driven methodology employs technology-independent software abstractions, e.g., in UML, to support diverse implementation platforms and enable reuse and automation.

• System of System (SoS) modeling theory goes beyond UML/MDA to clarify issues in integration, interoperability, composability, and reuse.

• SoS theory concepts include spectrum of integration/interoperability modes, linguistic levels of component interaction, and pragmatic frames.

• DEVS system theory –based framework operationalizes SoS theory and supports automated development and rigorous testing in realistic GIG/SOA environments.
Books and Web Links

www.acims.arizona.edu

www.sce.carleton.ca/faculty/wainer/index.html

RTSync.com
More Demos and Links
http://www.acims.arizona.edu/demos/demos.shtml

- Integrated Development and Testing Methodology:
  - AutoDEVS (ppt) & DEMO
    - Natural language-based Automated DEVS model generation
    - BPMN/BPEL-based Automated DEVS model generation
    - Net-centric SOA Execution of DEVS models
    - DEVS Unified Process for Integrated Development and Testing of SOA

- Intrusion Detection System on DEVS/SOA
BACKUP
Example: Learning a sequence of actions

- Problem: learn a skill that requires a sequence of actions; assume the sequence of activation is known, and the actions exist as alternative components for selection – the right component has to be selected for each step. For simplicity, assume there are two actions, right and wrong, for each step.
- Example: move an object from one place to another – 1st step: move grabber to right place vs move to another place; 2nd step: lift the object vs drop the object; 3rd step: move the grabber to the target location, etc.
- You must get each step right before learning the next.
- Example: A skill has 4 steps. A coupled model is shown next that represents the components and coupling, where each component can be either right or wrong.
- Step1 starts and if it is right, triggers Step2; if Step1 is wrong, then the trial is over and a new candidate is generated. If Step2 is right, it triggers Step3, etc.
- Each right step sends an output to the Sum (experimental frame).
Sequential Activation

Step 1 initiates action – it needs to be right to activate Step 2 and to add 1 to the sum.

Step 2 continues action – it needs to be right to activate Step 3 and to add 1 to the sum – and so on.

Maximum score is 4 if all steps completed – get partial score of N for N steps completed.
Example: Learning a sequence of actions

- We represent such a situation as an SES, with steps as components, each specialized by right and wrong.
- The SES also has the coupling that applies to any coupled model that can be pruned.

**Natural Language Specification**

From the message perspective, sequentialActivation is made of firstStep, secondStep, thirdStep, fourthStep, and sum!

firstStep can be wrong or right in correctness!
secondStep can be wrong or right in correctness!
thirdStep can be wrong or right in correctness!
fourthStep can be wrong or right in correctness!

From the message perspective, firstStep sends out to sum as in!
From the message perspective, secondStep sends out to sum as in!
From the message perspective, thirdStep sends out to sum as in!
From the message perspective, fourthStep sends out to sum as in!

From the message perspective, firstStep sends out to secondStep as in!
From the message perspective, secondStep sends out to thirdStep as in!
From the message perspective, thirdStep sends out to fourthStep as in!
Learning Sequential Activation = Pruning the SES

For a pruned SES, right or wrong is selected for each step at random.

For maximum score, right must be selected for each step.
Pruning with Biasing Using Component Ratings/Achievement

evaluate generated compositions (pruned SES) by sum of achievement of components

evaluateCompositions

transform

best estimate

pruneAllSpecsUsingEstimatedBest

sequentialActivation

thirdStep

fourthStep

thirdSpec

fourthSpec

wrong

right

wrong

right

wrong

right

wrong

right

wrong

right

wrong

right
Analysis of Accumulated Credit for each Right Step

Each right step accumulates its own score plus sum of the downstream right step scores (since it participates in them).

With 1/8 probability the activation ends after right step 2, and the score is then

Actual measured credit accumulation for long enough run

<table>
<thead>
<tr>
<th>Right step</th>
<th>Predicted ratio</th>
<th>Measured ratio</th>
<th>Measured ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.75</td>
<td>.30</td>
<td>3.75</td>
</tr>
<tr>
<td>2</td>
<td>2.75</td>
<td>.22</td>
<td>2.75</td>
</tr>
<tr>
<td>3</td>
<td>1.75</td>
<td>.14</td>
<td>1.75</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>.08</td>
<td>1</td>
</tr>
</tbody>
</table>
Analysis of Accumulated Credit for each Wrong Step

Each wrong step accumulates the previous step score.

With $\frac{1}{8}$ probability the activation ends after right step 2, and the score is then 2.

<table>
<thead>
<tr>
<th>Wrong step</th>
<th>Predicted ratio</th>
<th>Measured ratio</th>
<th>Measured ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1.33</td>
<td>.04</td>
<td>1.33</td>
</tr>
<tr>
<td>3</td>
<td>1.33</td>
<td>.04</td>
<td>1.33</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>.03</td>
<td>1</td>
</tr>
</tbody>
</table>

Actual measured credit accumulation for long enough run.
Pragmatic Frame: Information Exchange Framework

Event Occurrence

World/Environment

Produce

Observe/capture the world state/change

SES Ontology

Pragmatic Frame

Filtering

Sub SES Ontology

Consumer

Message

Data

Describe current data

Describe the world state/change
Interoperability Problems in Complex Endeavors

Example: Disaster Relief

Common Goal: Coordinate the distribution of food to the disaster area

Major Bottleneck: Participants have different resources and use different terminologies and business processes to deliver food

Solution: Provide effective coordination via interoperability afforded by pragmatic frame + probabilistic ontology concepts
Pragmatic Frame + Probabilistic Ontology Approach

Probabilistic Ontologies provide interface satisfying specific requirements of food distribution Frame

Frame specifies requirements for knowledge integration wrt Food distribution

Local Gov’t Agency

Probabilistic Ontology

Humanitarian NGO

Probabilistic Ontology

Pragmatic Frame: Food Distribution

Pragmatic Frame: Shelter Location

Military Agency

Probabilistic Ontology

International Gov’t Agency

Probabilistic Ontology