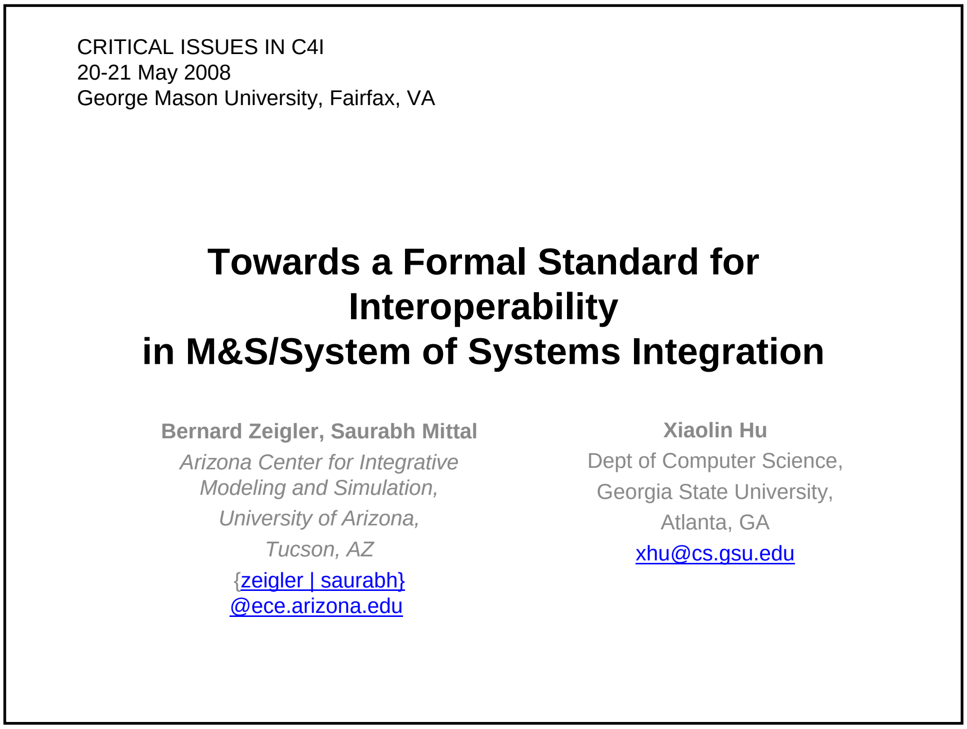
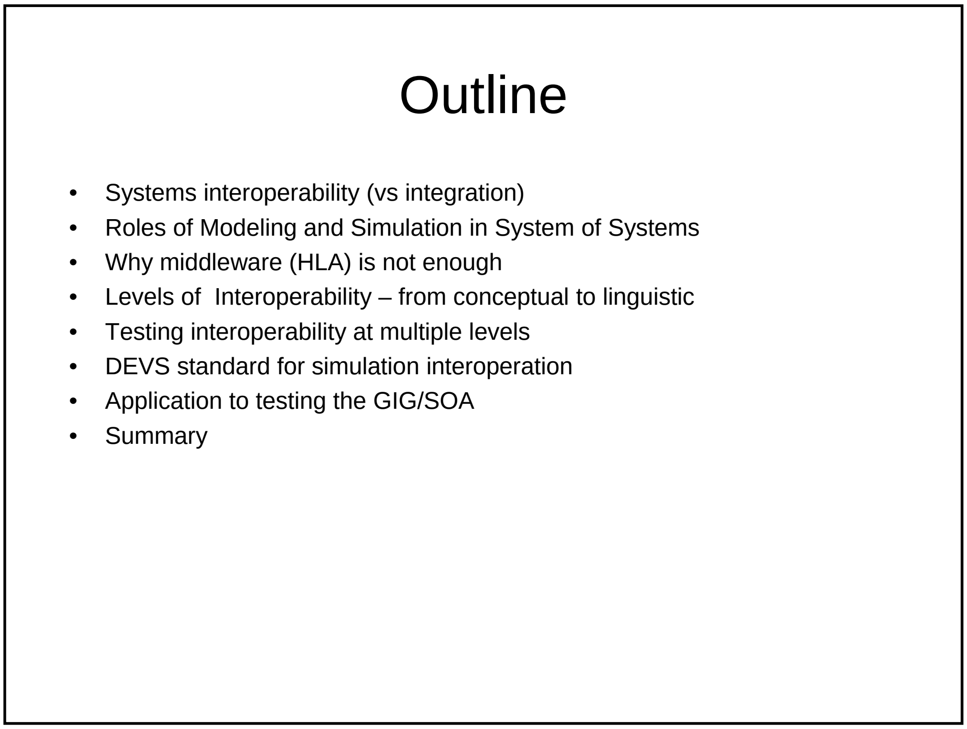


CRITICAL ISSUES IN C4I
20-21 May 2008
George Mason University, Fairfax, VA

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

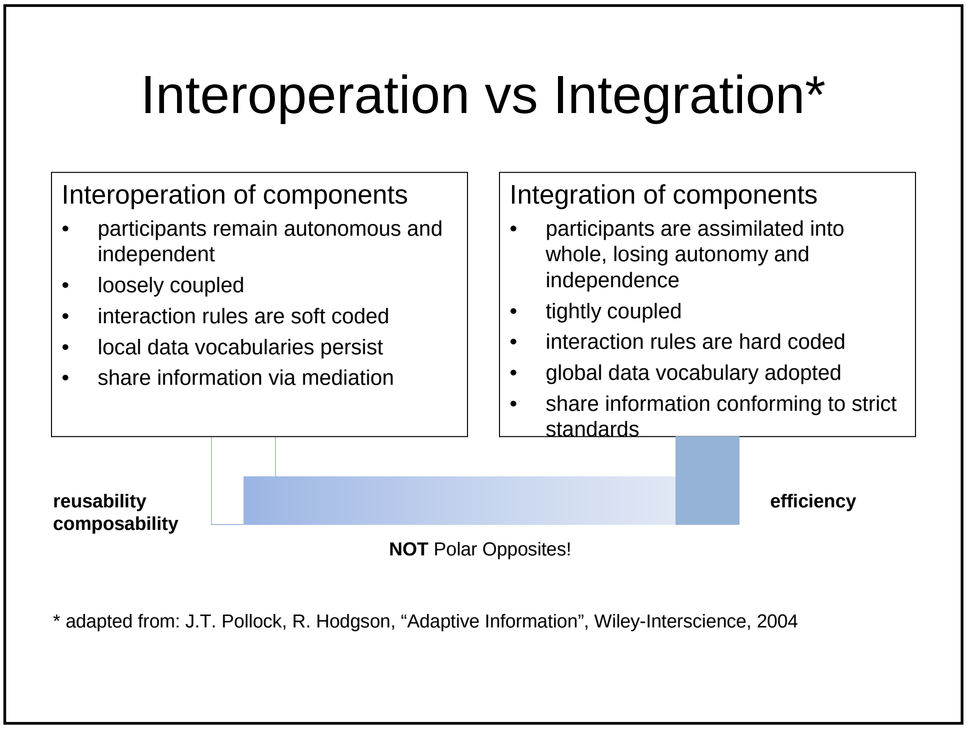
Bernard Zeigler, Saurabh Mittal
Arizona Center for Integrative Modeling and Simulation,
University of Arizona,
Tucson, AZ
{zeigler | saurabh}
@ece.arizona.edu

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

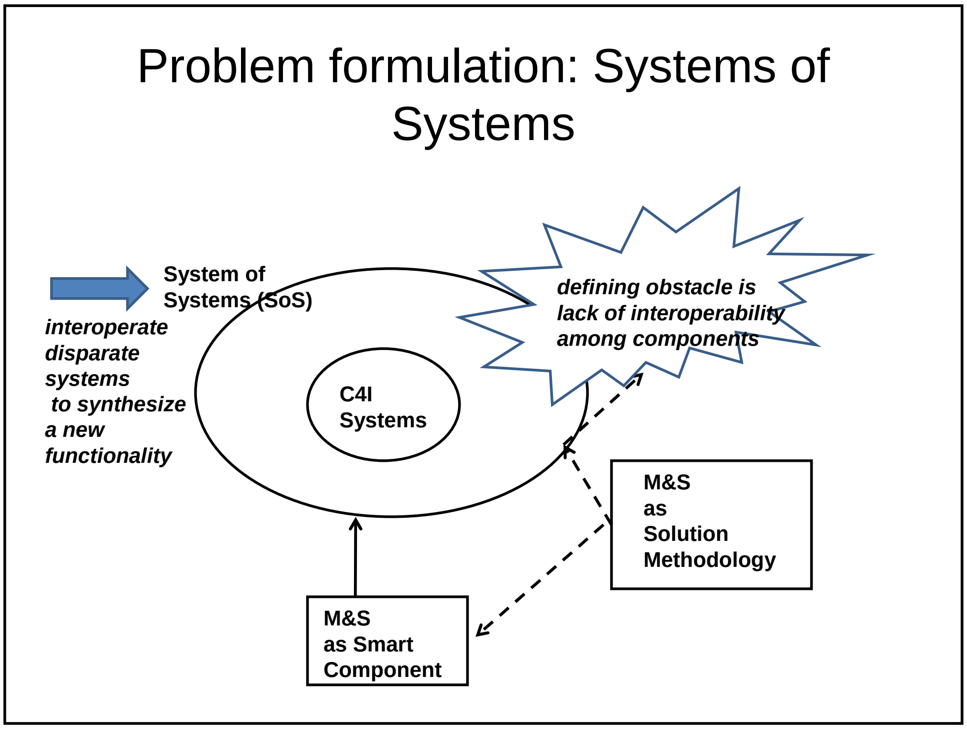
- participants remain autonomous and independent
- loosely coupled
- interaction rules are soft coded
- local data vocabularies persist
- share information via mediation

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

NOT Polar Opposites!

* adapted from: J.T. Pollock, R. Hodgson, “Adaptive Information”, Wiley-Interscience, 2004



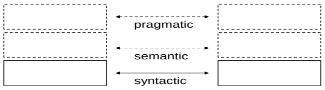
Problem formulation: Systems of Systems

System of Systems (SoS)
interoperate disparate systems to synthesize a new functionality
C4I Systems
defining obstacle is lack of interoperability among components
M&S as Solution Methodology
M&S as Smart Component

Tolk’s Levels of Conceptual Interoperability Model

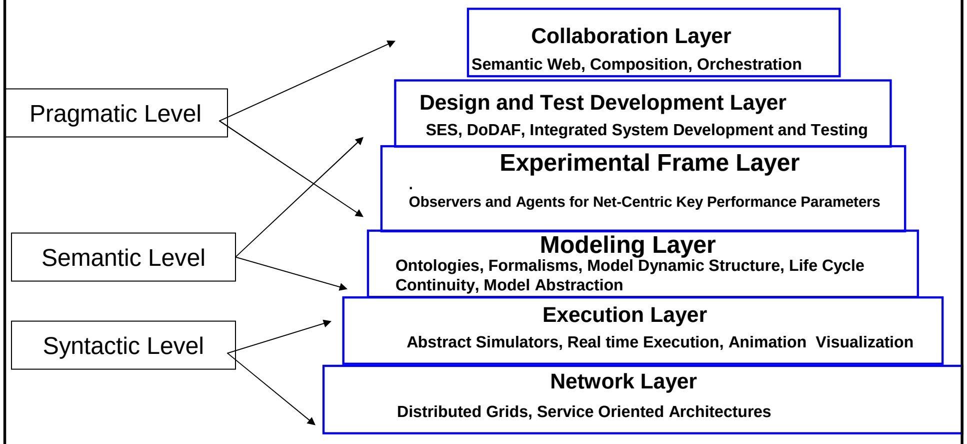
Level of Conceptual Interoperability	Characteristic	Key Condition
Conceptual	The assumptions and constraints underlying the meaningful abstraction of reality are aligned	Requires that conceptual models be documented based on engineering methods enabling their interpretation and evaluation by other engineers.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

Linguistic Levels of Interoperability



Linguistic Level	Interoperability Demonstrated if:	Example
Pragmatic – How information in message is used	The receiver reacts to the message in a manner that the sender intends	A commander’s order is obeyed by the troops in the field as the commander intended. (This assumes semantic interoperability.)
Semantic – Shared understanding of meaning of messages	The receiver assigns the same meaning as the sender did to the message.	An order from a commander to multi-national participants in a coalition operation is understood in the same manner despite translation into different languages.
Syntactic – Common rules governing composition and transmitting of messages	The consumer is able to receive and parse the sender’s message	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.

Mapping M&S Layers to Linguistic Levels



Pragmatic Level

Semantic Level

Syntactic Level

Collaboration Layer
Semantic Web, Composition, Orchestration

Design and Test Development Layer
SES, DoDAF, Integrated System Development and Testing

Experimental Frame Layer
.
Observers and Agents for Net-Centric Key Performance Parameters

Modeling Layer
Ontologies, Formalisms, Model Dynamic Structure, Life Cycle Continuity, Model Abstraction

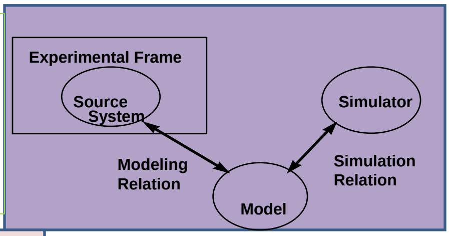
Execution Layer
Abstract Simulators, Real time Execution, Animation Visualization

Network Layer
Distributed Grids, Service Oriented Architectures

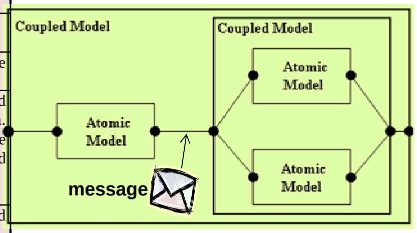
Background: DEVS M&S Framework

Discrete Event Systems Specification (DEVS)

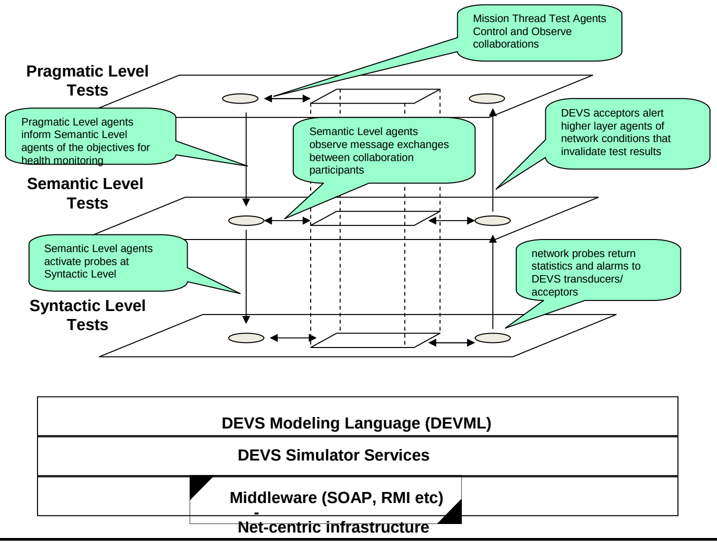
- Based on mathematical formalism using system theoretic principles
- Separation of Model, Simulator and Experimental Frame
- Atomic and Coupled types
- Hierarchical modular composition



Level	Name	System Specification at this level
4	Coupled Systems	System built from component systems with coupling recipe.
3	I/O System Structure	System with state and state transitions to generate the behavior.
2	I/O Function	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.
1	I/O Behavior	Collection of input/output pairs constituting the allowed behavior of the system from an external Black Box view.
0	I/O Frame	Input and output variables and ports together with allowed values.



DEVS Modeling and Simulation Infrastructure supports simultaneous testing at multiple levels



Mission Thread Test Agents Control and Observe collaborations

Pragmatic Level Tests

Pragmatic Level agents inform Semantic Level agents of the objectives for health monitoring

Semantic Level agents observe message exchanges between collaboration participants

DEVS acceptors alert higher layer agents of network conditions that invalidate test results

Semantic Level Tests

Semantic Level agents activate probes at Syntactic Level

network probes return statistics and alarms to DEVS transducers/ acceptors

Syntactic Level Tests

DEVS Modeling Language (DEVML)

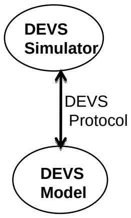
DEVS Simulator Services

Middleware (SOAP, RMI etc)
-
Net-centric infrastructure

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:

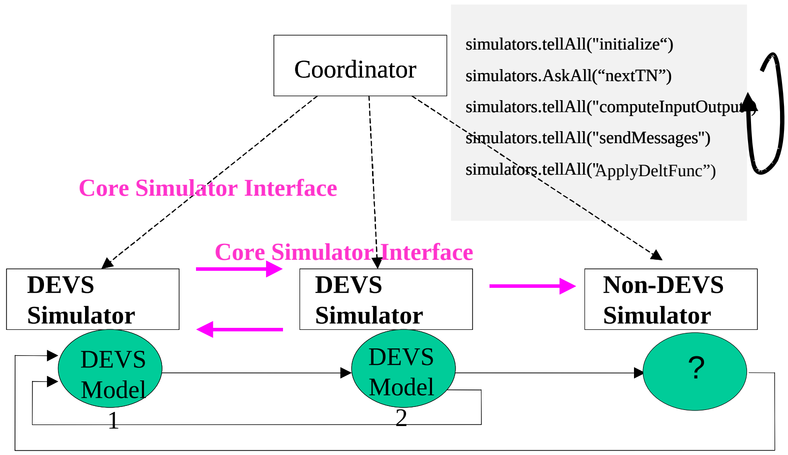


- *declaring who takes part in the simulation:*
  - o format for referencing federates (participants)
- *declaring how federates exchange information:*
  - o format for their message exchange patterns

- *executing an iterative cycle that*
  - *controls how time advances:*
    - o updating the clock based on next event times
  - *determines when federates exchange messages:*
    - o the point in the cycle when all interchange takes place
  - *determines when federates do internal state updating*
    - o the point in the cycle when next event times are collected

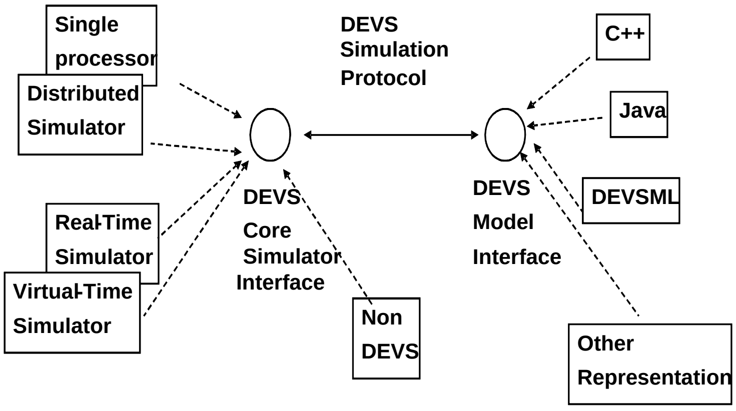
Note:
*If the federates are DEVS compliant then the simulation is **provably correct** in the sense that the DEVS closure under coupling theorem guarantees a well-defined resulting structure and behavior.*

DEVS Simulation Protocol



```
simulators.tellAll("initialize“)
simulators.AskAll(“nextTN”)
simulators.tellAll("computeInputOutput")
simulators.tellAll("sendMessages")
simulators.tellAll(‘ApplyDeltFunc”)
```

Concept of DEVS Standard



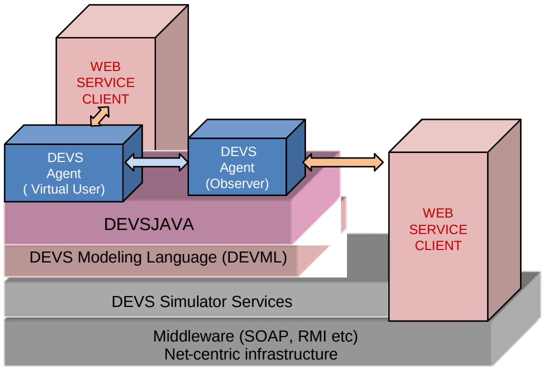
Core Simulator Interface

```
simulatorstellAll("initialize“)
simulatorsAskAll(“nextTN”)
simulatorstellAll("computeInputOutput”)
simulatorstellAll("sendMessages")
simulatorstellAll("ApplyDeltFunc”)
```

```
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

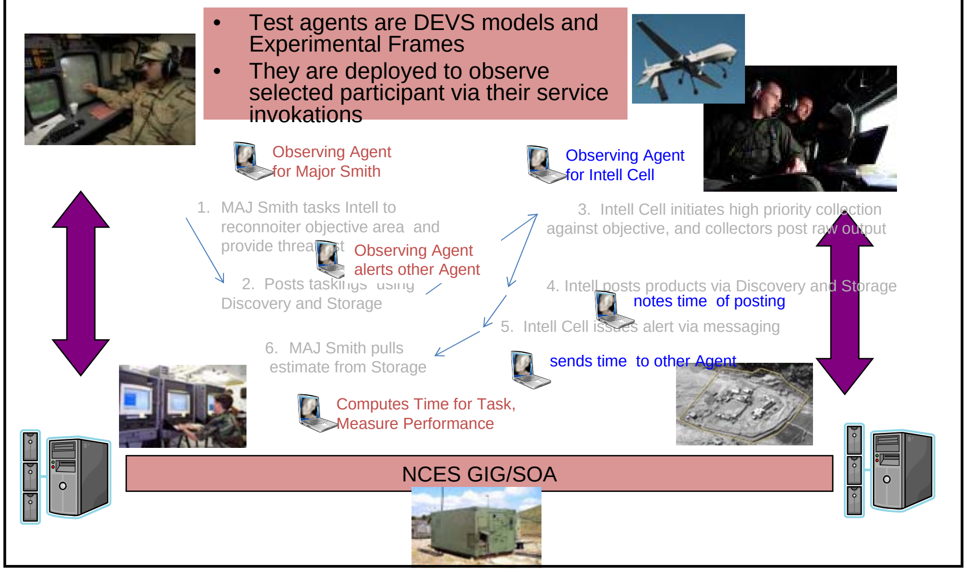
DEVS/SOA Infrastructure: Supports Deployment and Execution of DEVS Models on the Web


WEB SERVICE CLIENT
DEVS Agent (Virtual User)
DEVS Agent (Observer)
WEB SERVICE CLIENT
DEVSJAVA
DEVS Modeling Language (DEVML)
DEVS Simulator Services
Middleware (SOAP, RMI etc)
Net-centric infrastructure

- **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**

Run Example

DEVS/SOA Infrastructure for GIG Mission Thread Testing



- Test agents are DEVS models and Experimental Frames
- They are deployed to observe selected participant via their service invokations

Observing Agent for Major Smith

Observing Agent for Intell Cell

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

Observing Agent alerts other Agent

2. Posts taskings using Discovery and Storage

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

4. Intell posts products via Discovery and Storage

notes time of posting

5. Intell Cell issues alert via messaging

6. MAJ Smith pulls estimate from Storage

sends time to other Agent

Computes Time for Task, Measure Performance

NCES GIG/SOA

Summary

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

DEVS-to-DEVS Interoperability

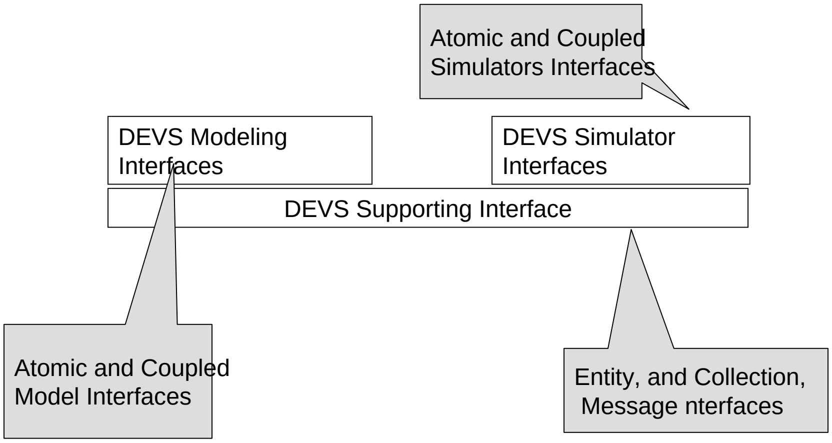
- DEVS standard facilitates interoperability at the syntactic, semantic and pragmatic levels

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

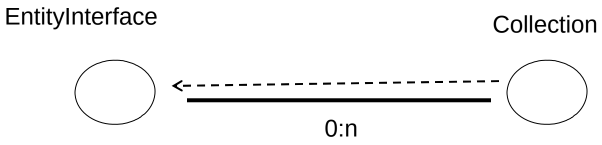
DEVS Simulator
Interfaces

DEVS Supporting Interface

Atomic and Coupled
Model Interfaces

Entity, and Collection,
Message nterfaces

DEVS Supporting Interfaces

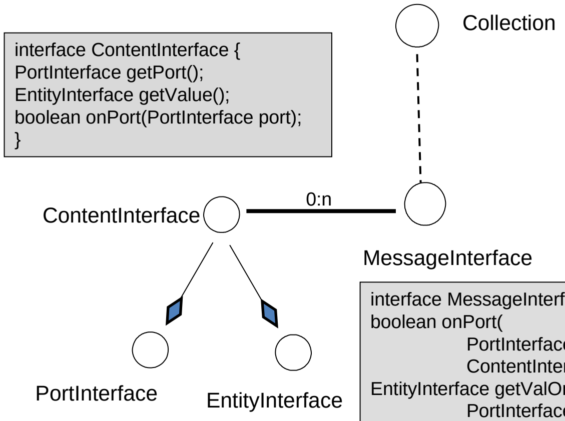


```
interface EntityInterface{
String getName();
boolean equalName(String name);
}
```

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

Message-related interfaces

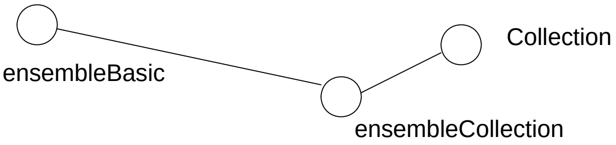
```
interface ContentInterface {
PortInterface getPort();
EntityInterface getValue();
boolean onPort(PortInterface port);
}
```



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

```
interface PortInterface
              extends
EntityInterface{
}
```

Ensemble Interfaces

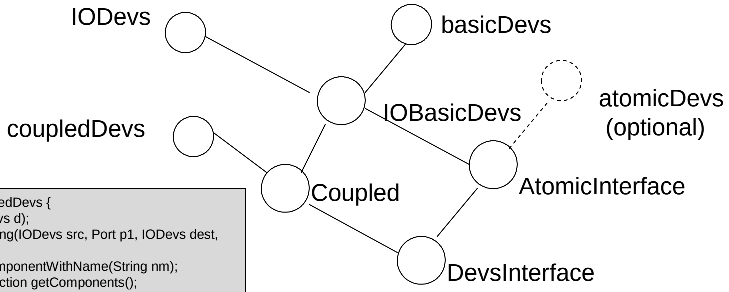


```
interface ensembleBasic {
void tellAll(Method m, EntityInterface[ ] args);
ensembleCollection askAll(Method m);
ensembleCollection which(Method m);
EntityInterface whichOne(Method m);
}

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

DEVS Model Interfaces

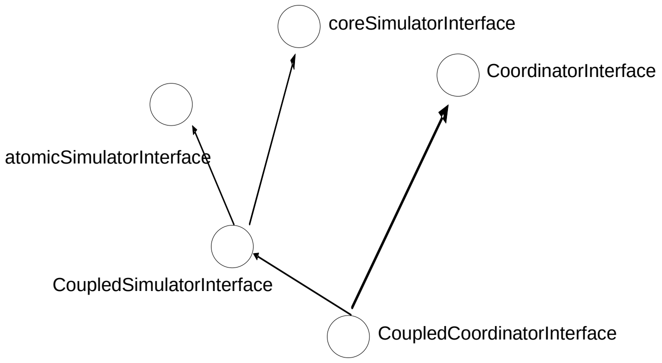
```
interface IODevs {
void addInport(String portName);
void addOutport(String portName);
ensembleCollection getInports();
ensembleCollection getOutports();
ContentInterface makeContent(PortInterface
port,EntityInterface value);
boolean   messageOnPort(MessageInterface x,
PortInterface port, ContentInterface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

DEVS Simulator Interfaces



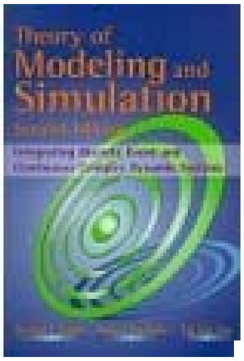
See also

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

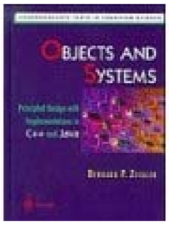
On

http://osa.inria.fr/wiki/NCMS/NCMS

Books and Web Links



MODELING AND SIMULATION-BASED DATA ENGINEERING
Introducing Pragmatics into Ontologies for Net-Centric Information Exchange
Bernard Zeigler
Phillip E. Hammonds
MODELING AND SIMULATION-BASED DATA ENGINEERING



devsworld.org

acims.arizona.edu

Rtsync.com