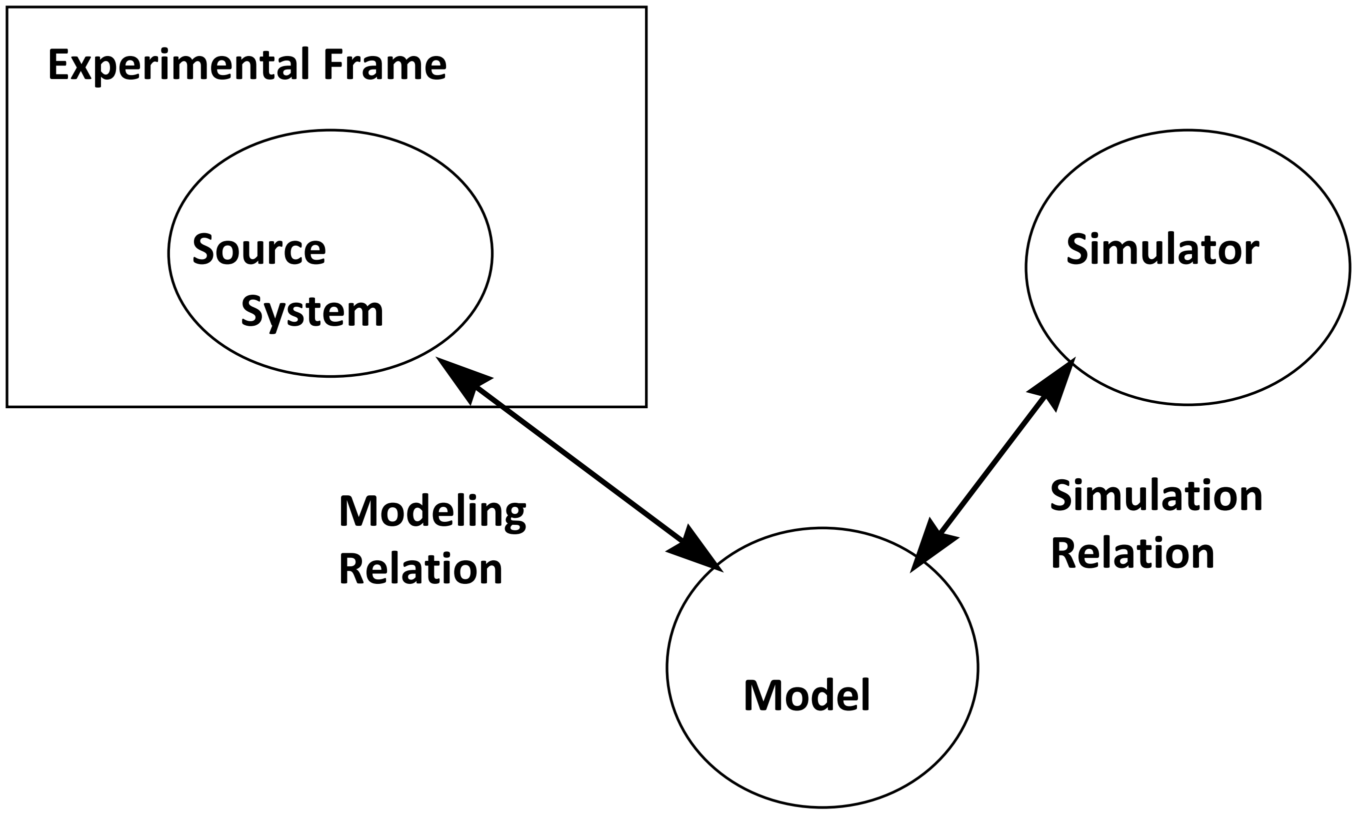
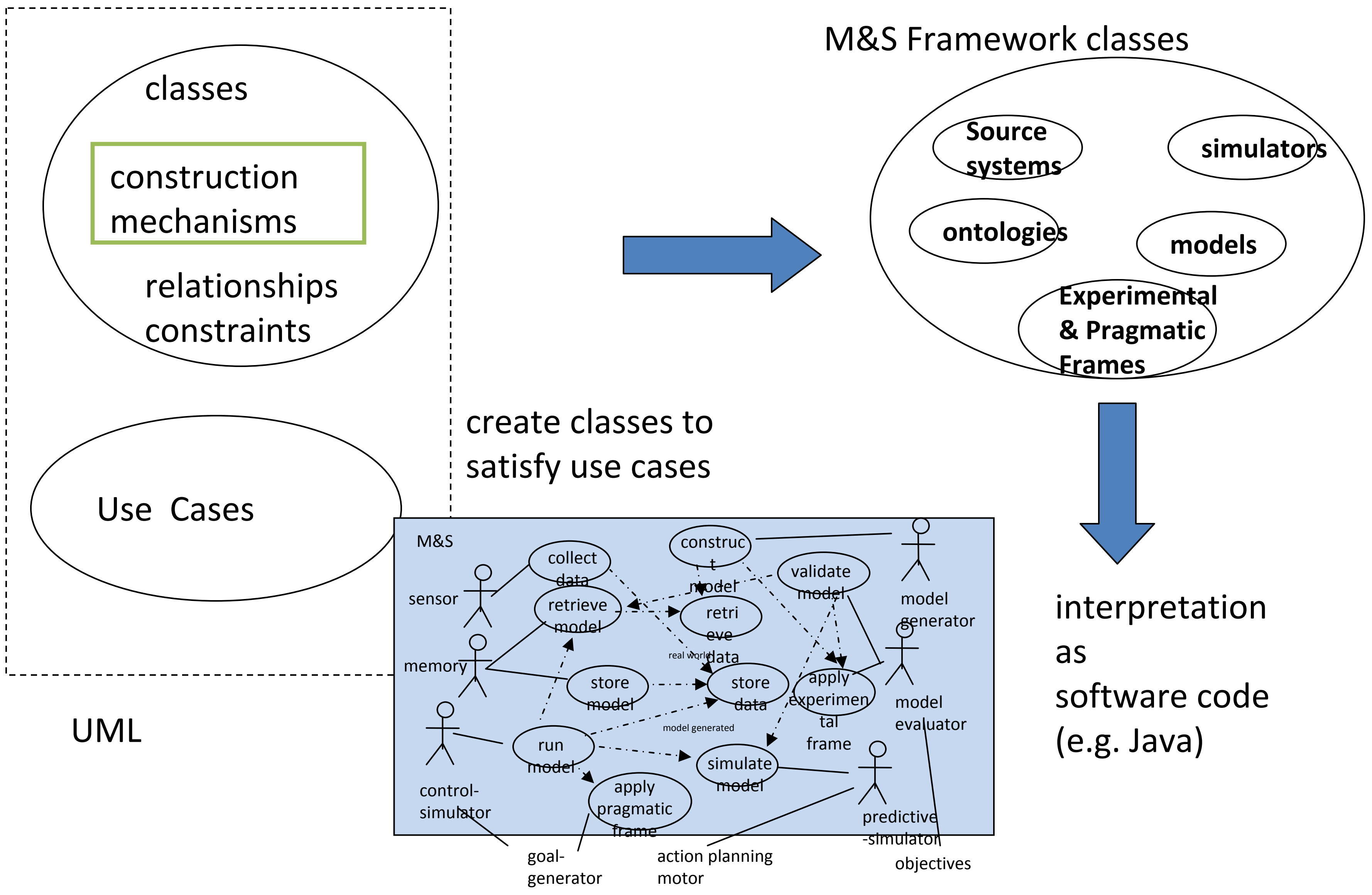
DEVS and SES as a Framework for Modeling and Simulation Tool Development

Bernard P. Zeigler
Arizona Center for Integrative Modeling and Simulation
University of Arizona

M&S Framework



M&S Framework formulated within UML



DEVS – Formal Specification of a System

A discrete event system specification (DEVS) is a structure

$M=<X,S,Y,\delta_{int}, \delta_{ext,} \delta_{con}, \lambda,ta>$

where

X is the set of input values,

S is a set of states,

Y is the set of output values,

δ_{int} :S->S is the internal transition function,

$\delta_{ext,}$:Q$\times$X->S is the external transition function,

$\delta_{con,}$:Q$\times$X->S is the confluent transition function,

ta: S->$R^{+}_{0,\infty}$

Where

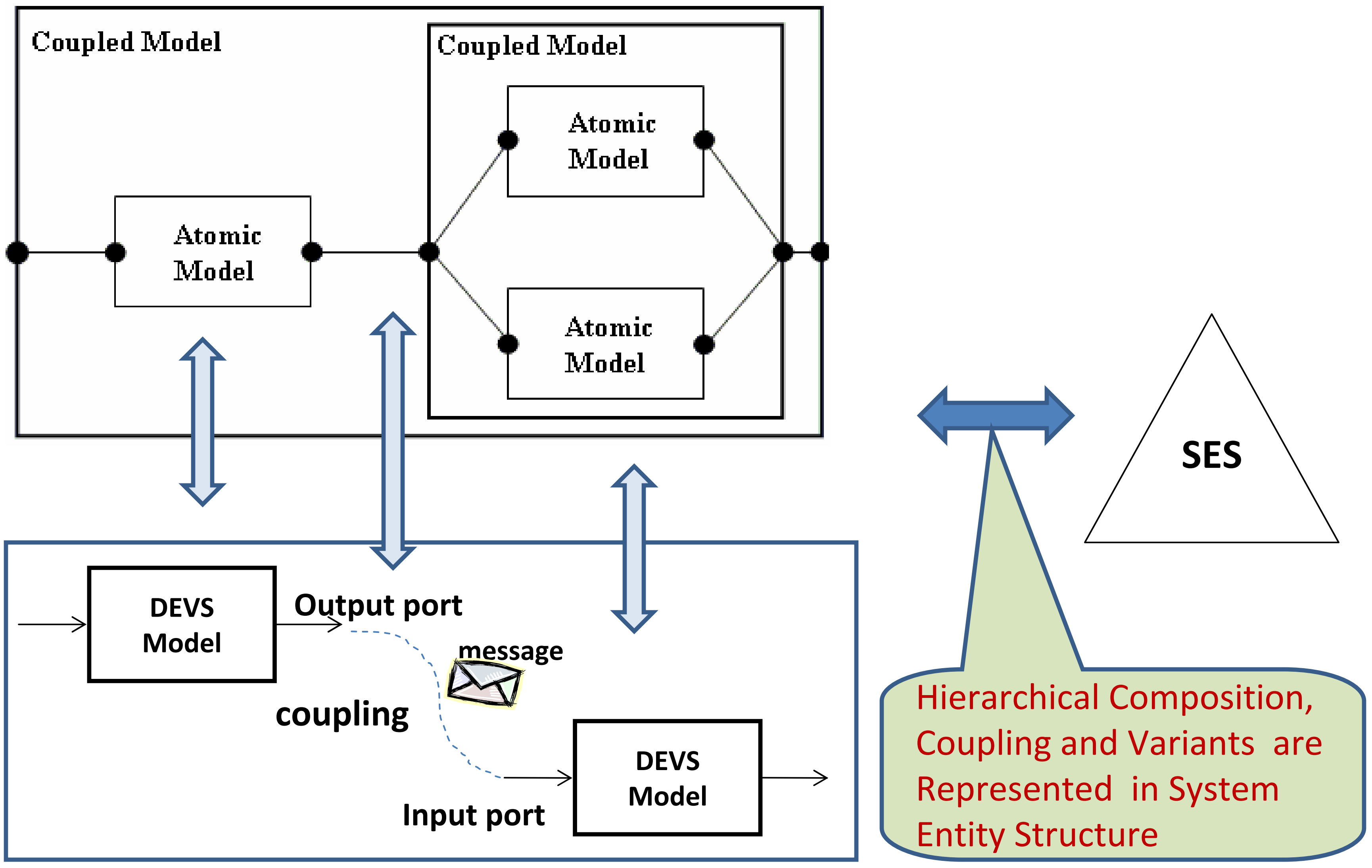
Q={(s,e)|s$\in$ S, 0 $\leq$e$\leq$ ta(s) } is the total state set,

e is the time elapsed since last transition,

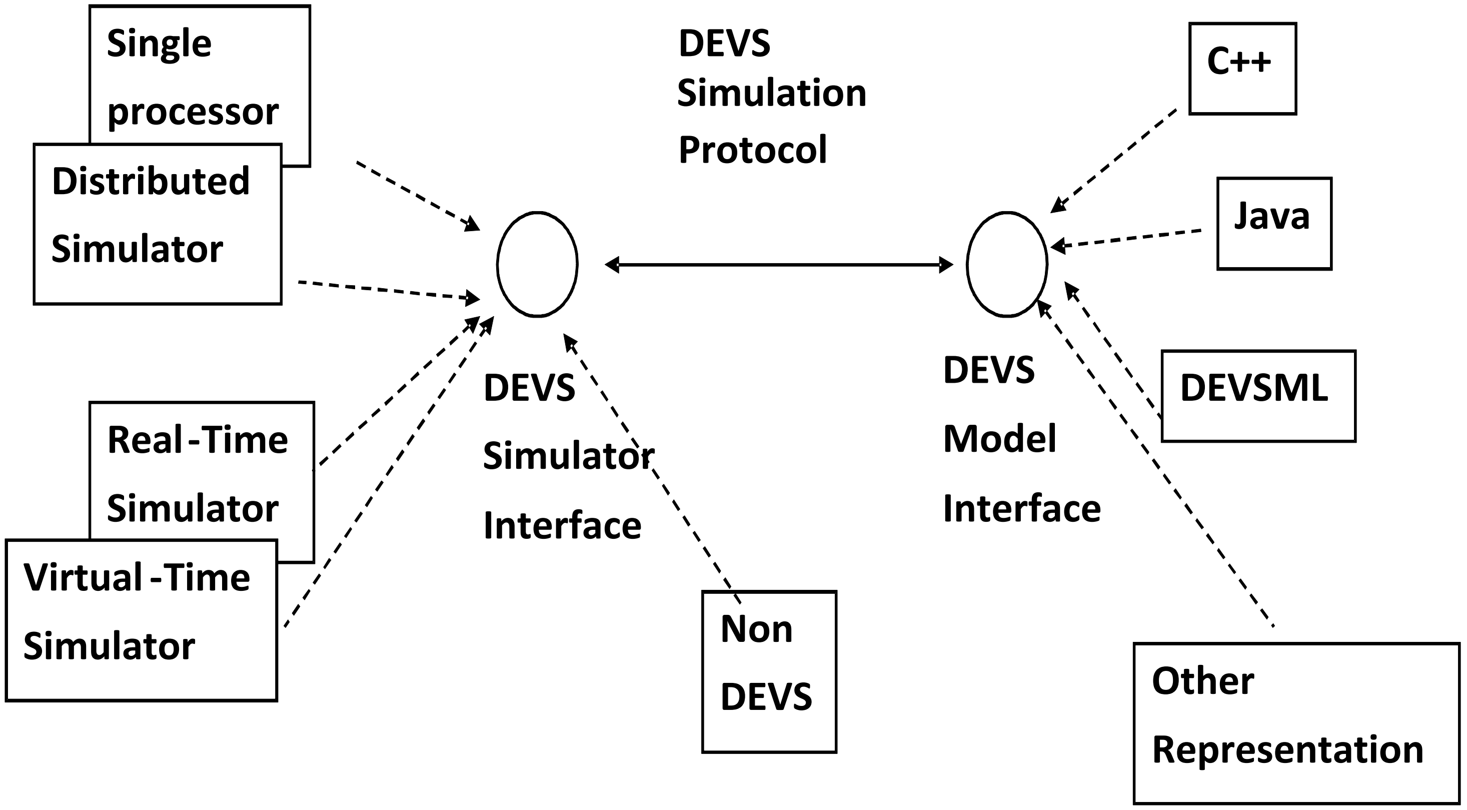
λ:S->Y is the output function and

$R^{+}_{0,\infty}$ is the set of positive reals with 0 and ∞

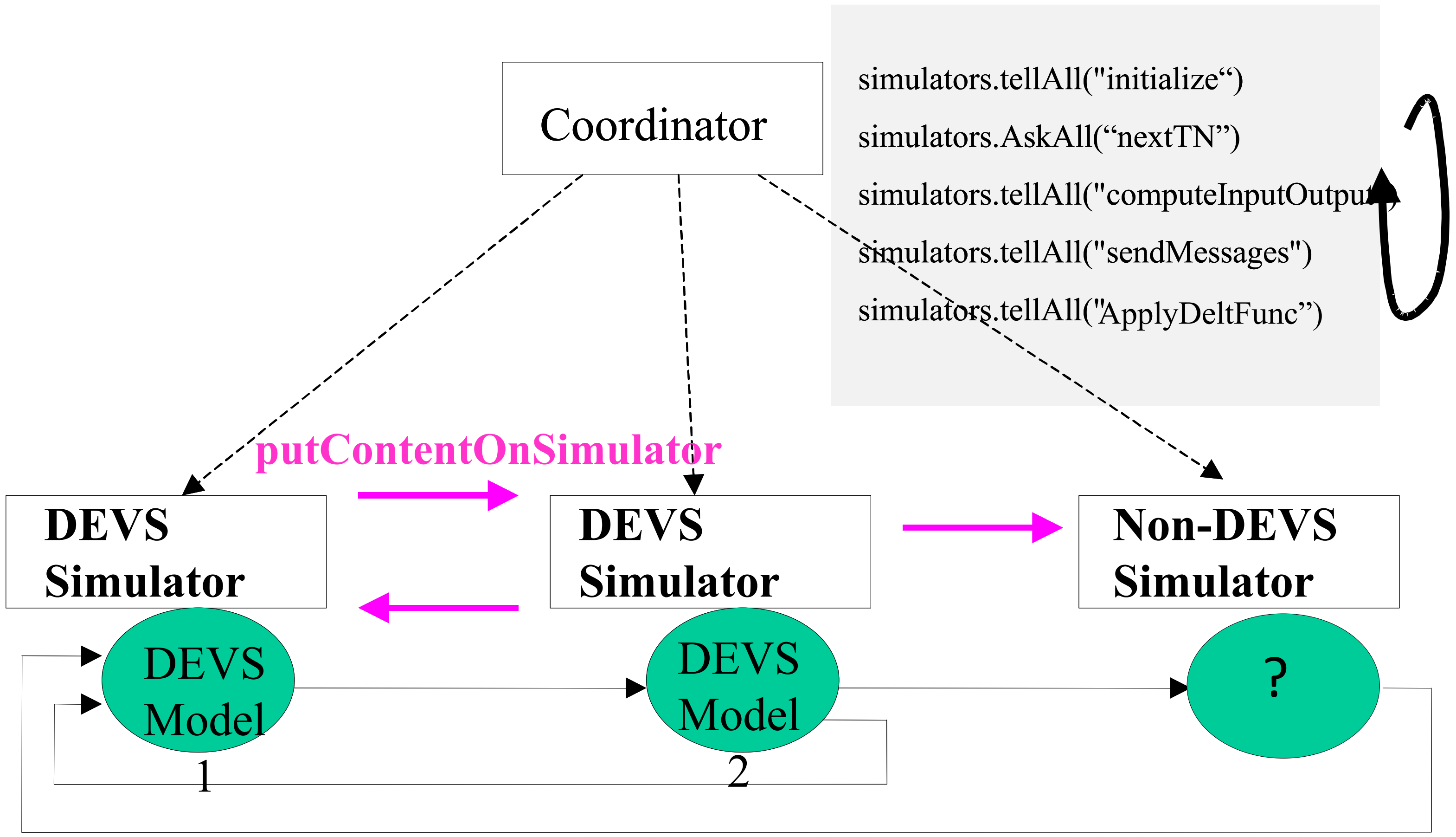
DEVS Hierarchical Modular Models



Concept of DEVS Standard



DEVS Simulation Protocol



Coordinator
simulators.tellAll("initialize“)
simulators.AskAll(“nextTN”)
simulators.tellAll("computeInputOutput")
simulators.tellAll("sendMessages")
simulators.tellAll("ApplyDeltFunc”)
putContentOnSimulator
DEVS Simulator
DEVS Simulator
Non-DEVS Simulator
DEVS Model
1
DEVS Model
2
?

Finite Deterministic DEVS : FD-DEVS

FDDEVS = <incomingMessageSet, outgoingMessageSet, StateSet, TimeAdvanceTable, InternalTransitionTable, ExternalTransitionTable,OutputTable>
where
incomingMessageSet, outgoingMessageSet, StateSet are finite sets
TimeAdvanceTable: StateSet → R0,∞+ (the positive reals with zero and infinity)
InternalTransitionTable: StateSet → StateSet
ExternalTransitionTable: StateSet × incomingMessageSet → StateSet, and
OutputTable: StateSet → 2outgoingMsgSet (= the set of subsets of outgoingMsgSet)

Natural Language For FDDEVS

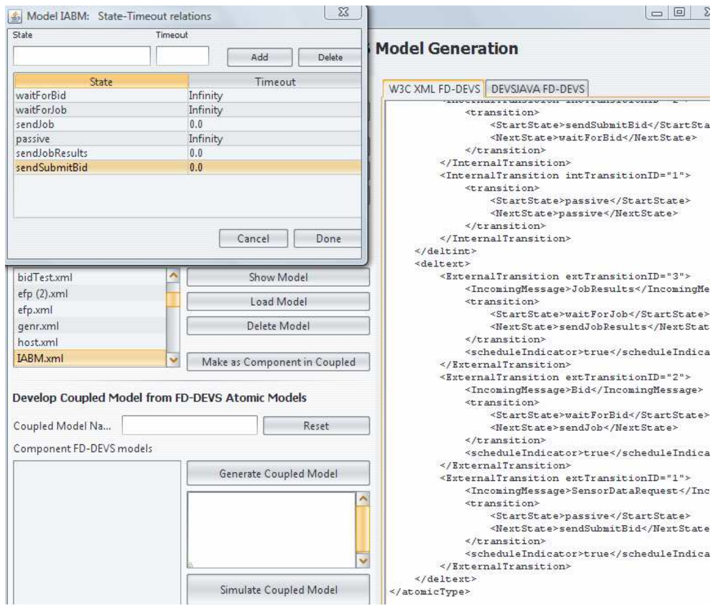
- *to start hold in PHASE for time SIGMA*
- *hold in PHASE for time SIGMA*
- *after PHASE then output MSG*
- *from PHASE go to PHASE’*
- *when in PHASE and receive MSG go to PHASE’*

<eventually>

Semantics defined by mapping into DEVS

FD-DEVS

- The “right” abstraction of DEVS – retains important timing properties
- Amenable to analysis
- Supports automation
- Maps to DEVSJAVA
- Supplies a skelton that can be extended to full DEVS
- Simple XML expression



Model IABM: State-Timeout relations

State	Timeout
waitForBid	Infinity
waitForJob	Infinity
sendJob	0.0
passive	Infinity
sendJobResults	0.0
sendSubmitBid	0.0

bidTest.xml
efp (2).xml
efp.xml
genr.xml
host.xml
IABM.xml

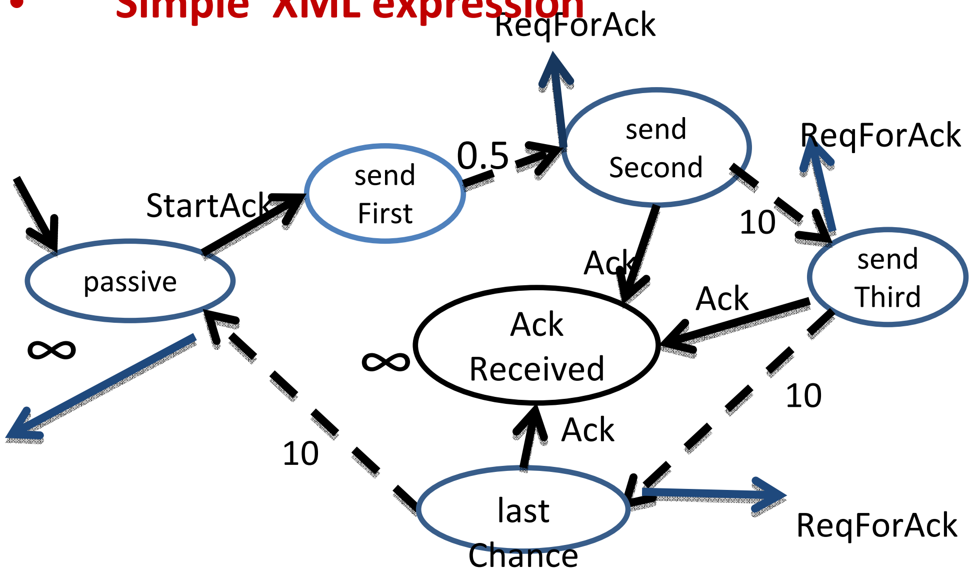
Model Generation

Develop Coupled Model from FD-DEVS Atomic Models

Coupled Model Na...

Component FD-DEVS models

```
            <transition>
                <StartState>sendSubmitBid</StartSta
                <NextState>waitForBid</NextState>
            </transition>
        </InternalTransition>
        <InternalTransition intTransitionID="1">
            <transition>
                <StartState>passive</StartState>
                <NextState>passive</NextState>
            </transition>
        </InternalTransition>
    </deltint>
    <deltext>
        <ExternalTransition extTransitionID="3">
            <IncomingMessage>JobResults</IncomingMe
            <transition>
                <StartState>waitForJob</StartState>
                <NextState>sendJobResults</NextStat
            </transition>
            <scheduleIndicator>true</scheduleIndica
        </ExternalTransition>
        <ExternalTransition extTransitionID="2">
            <IncomingMessage>Bid</IncomingMessage>
            <transition>
                <StartState>waitForBid</StartState>
                <NextState>sendJob</NextState>
            </transition>
            <scheduleIndicator>true</scheduleIndica
        </ExternalTransition>
        <ExternalTransition extTransitionID="1">
            <IncomingMessage>SensorDataRequest</Inc
            <transition>
                <StartState>passive</StartState>
                <NextState>sendSubmitBid</NextState
            </transition>
            <scheduleIndicator>true</scheduleIndica
        </ExternalTransition>
    </deltext>
</atomicType>
```



Example of Natural Language Spec:

1.to start passivate in passive

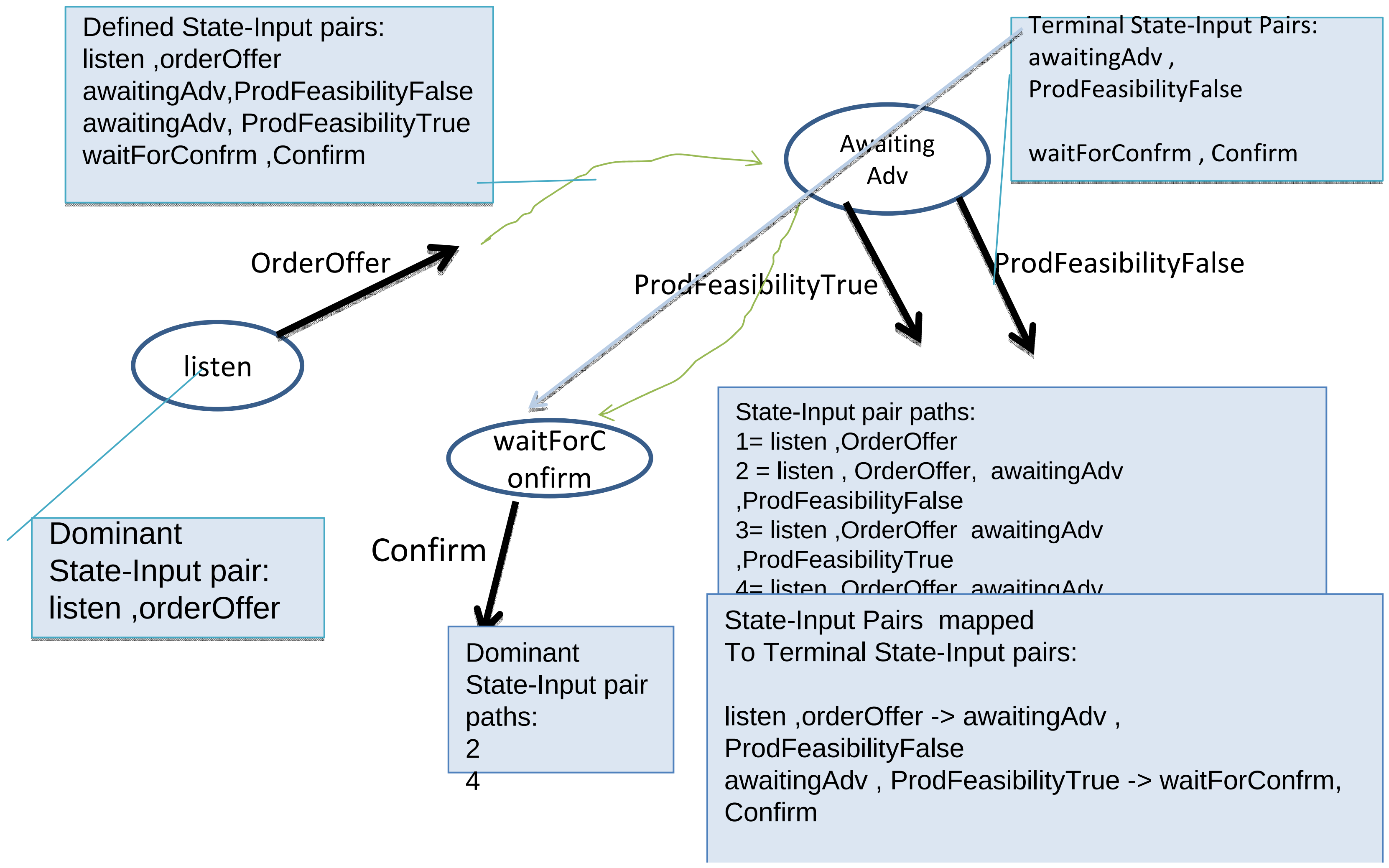
2.when in passive and receive StartAck go to sendFirst

3.hold in sendFirst for time 0.5 then output ReqForAck and go to sendSecond

4.hold in ackReceived for time Infinity

5.…

Automated Analysis: Based on State Input Pairs Enables Automated Test Model Generation



Defined State-Input pairs:
listen ,orderOffer
awaitingAdv,ProdFeasibilityFalse
awaitingAdv, ProdFeasibilityTrue
waitForConfrm ,Confirm

Terminal State-Input Pairs:
awaitingAdv ,
ProdFeasibilityFalse

waitForConfrm , Confirm

Dominant
State-Input pair:
listen ,orderOffer

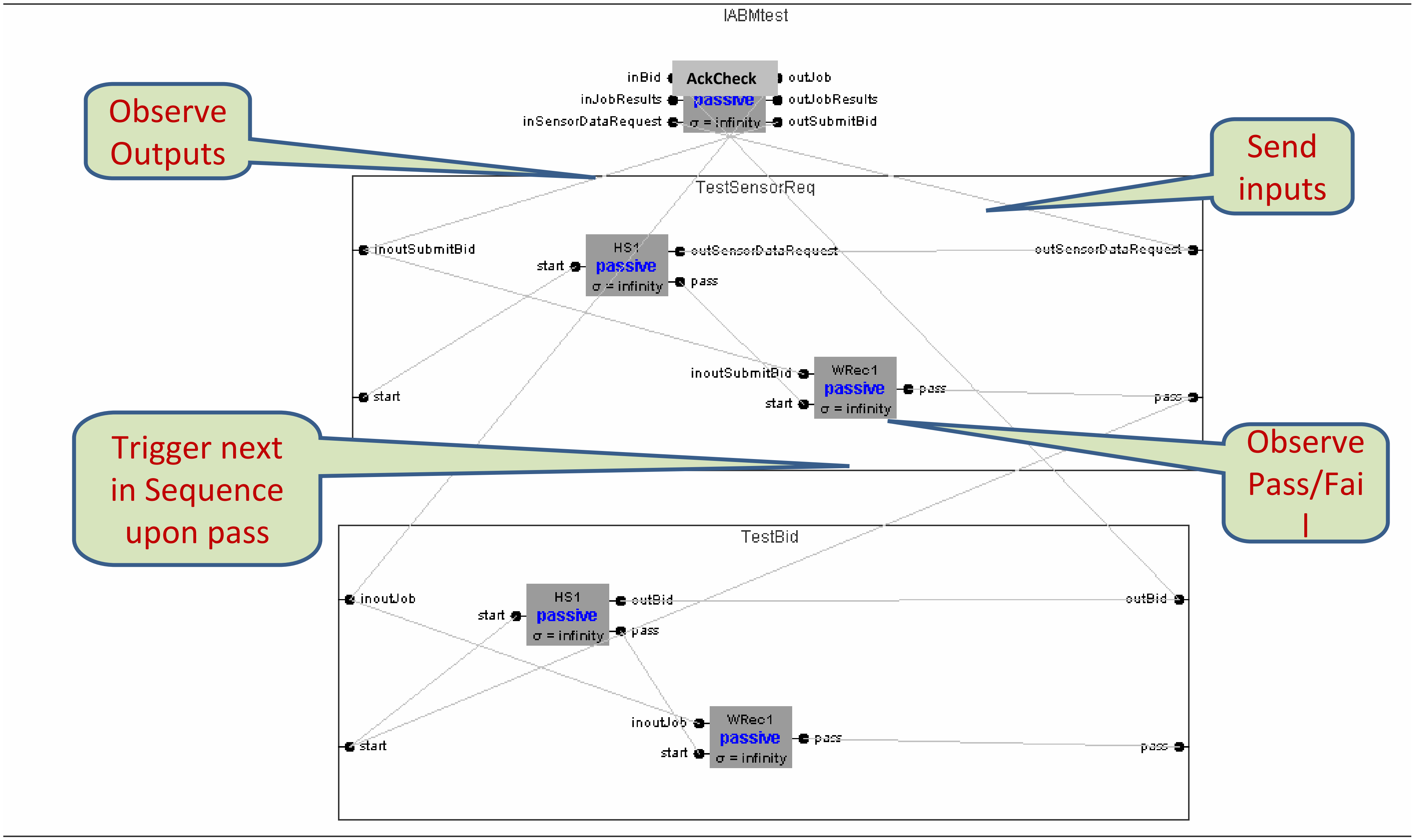
State-Input pair paths:
1= listen ,OrderOffer
2 = listen , OrderOffer, awaitingAdv ,ProdFeasibilityFalse
3= listen ,OrderOffer awaitingAdv ,ProdFeasibilityTrue
4= listen ,OrderOffer awaitingAdv

Dominant
State-Input pair paths:
2
4

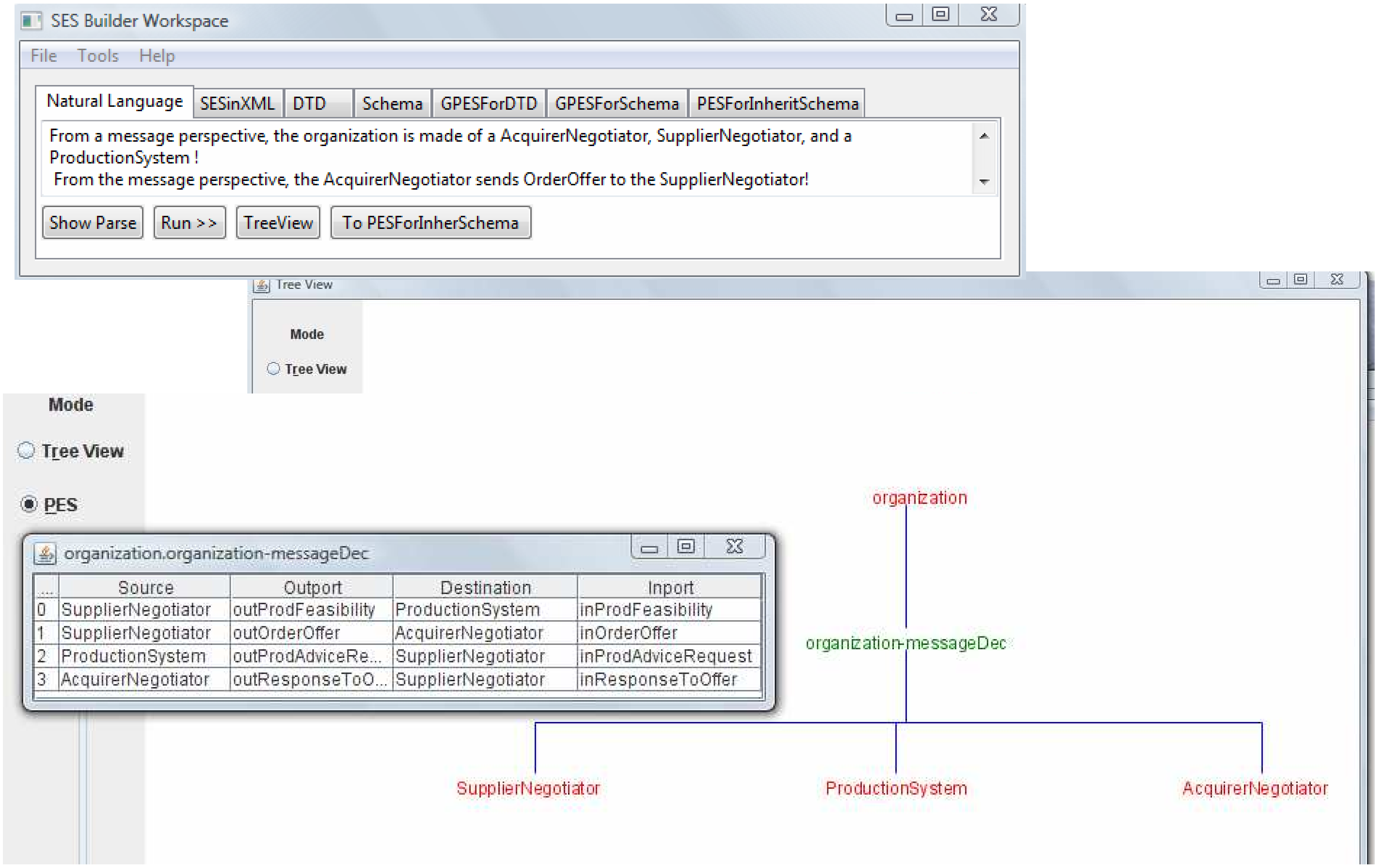
State-Input Pairs mapped
To Terminal State-Input pairs:

listen ,orderOffer -> awaitingAdv , ProdFeasibilityFalse
awaitingAdv , ProdFeasibilityTrue -> waitForConfrm, Confirm

Generated Test Models in DEVSJAVA SimView



System Entity Structure (SES) : SESBuilder

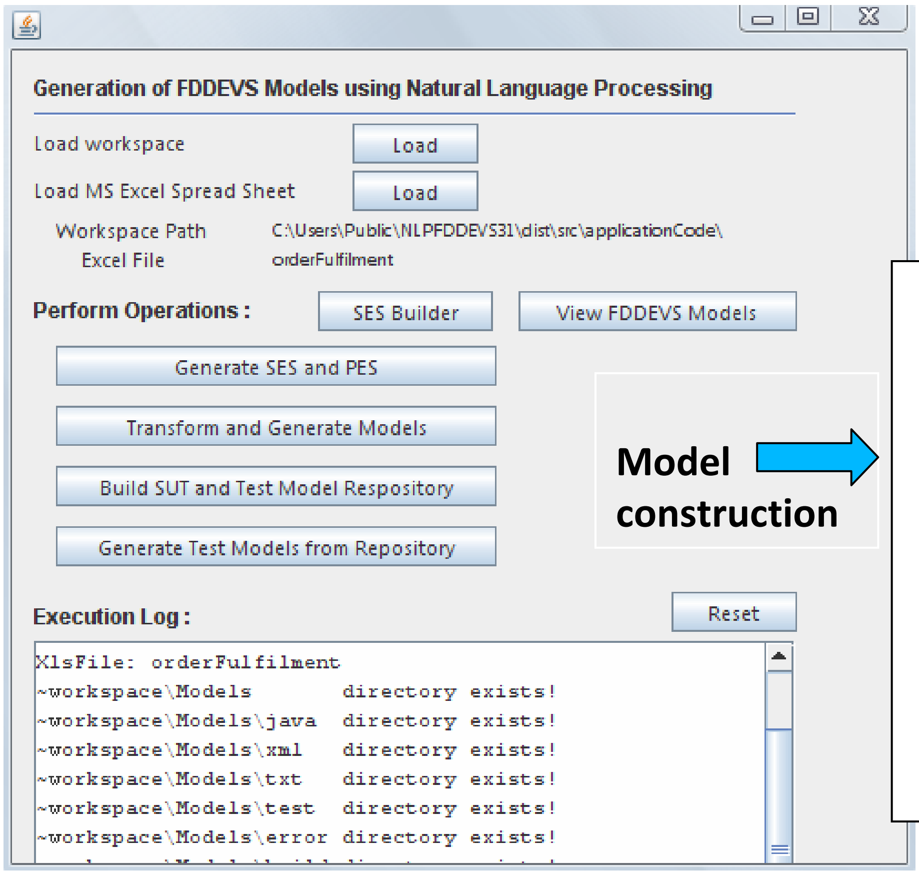


From a message perspective, the organization is made of a AcquirerNegotiator, SupplierNegotiator, and a ProductionSystem !
From the message perspective, the AcquirerNegotiator sends OrderOffer to the SupplierNegotiator!

organization.organization-messageDec

...	Source	Outport	Destination	Inport
0	SupplierNegotiator	outProdFeasibility	ProductionSystem	inProdFeasibility
1	SupplierNegotiator	outOrderOffer	AcquirerNegotiator	inOrderOffer
2	ProductionSystem	outProdAdviceRe...	SupplierNegotiator	inProdAdviceRequest
3	AcquirerNegotiator	outResponseToO...	SupplierNegotiator	inResponseToOffer

System Entity Structure/Model Base Repository: Support Automated DEVS Generation and Reuse



Generation of FDDEVS Models using Natural Language Processing

Load workspace	Load
Load MS Excel Spread Sheet	Load
Workspace Path	C:\Users\Public\NLPFDDEVS31\dist\src\applicationCode\
Excel File	orderFulfilment

Perform Operations : SES Builder · View FDDEVS Models

Generate SES and PES

Transform and Generate Models

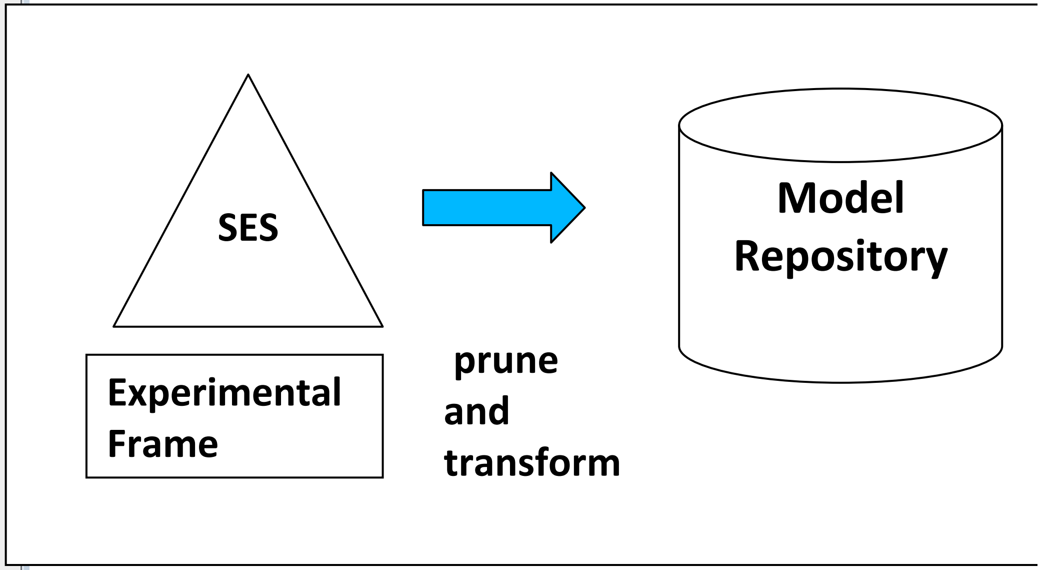
Build SUT and Test Model Respository

Generate Test Models from Repository

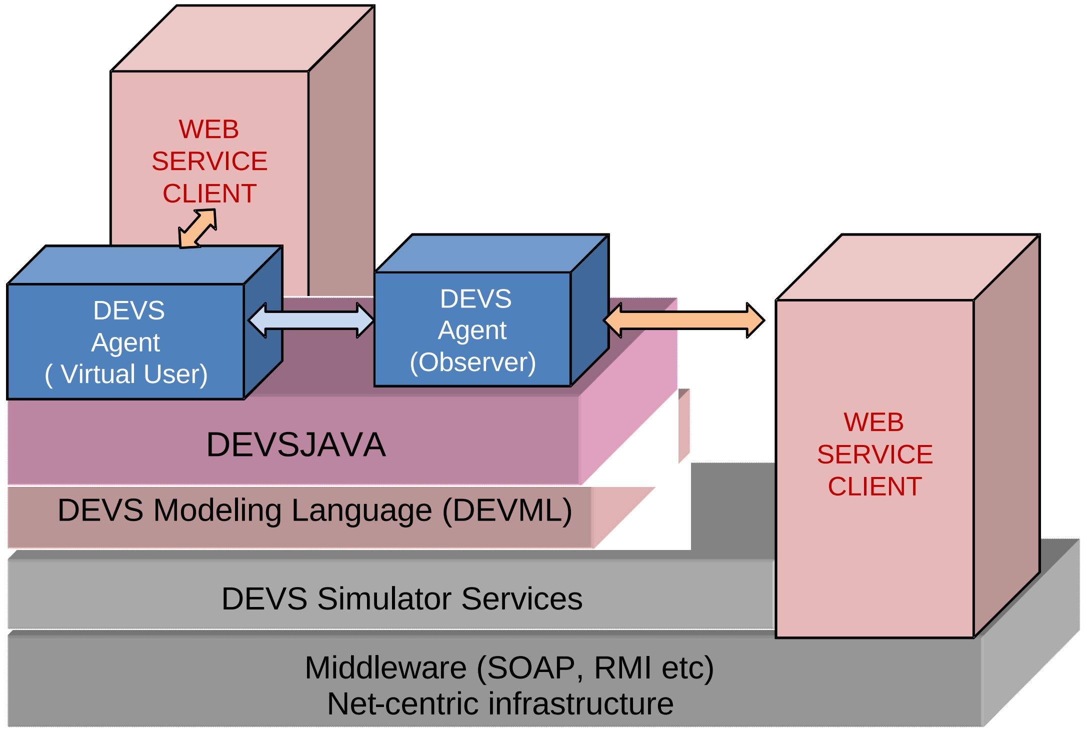
Execution Log : Reset

```
XlsFile: orderFulfilment
~workspace\Models       directory exists!
~workspace\Models\java  directory exists!
~workspace\Models\xml   directory exists!
~workspace\Models\txt   directory exists!
~workspace\Models\test  directory exists!
~workspace\Models\error directory exists!
```

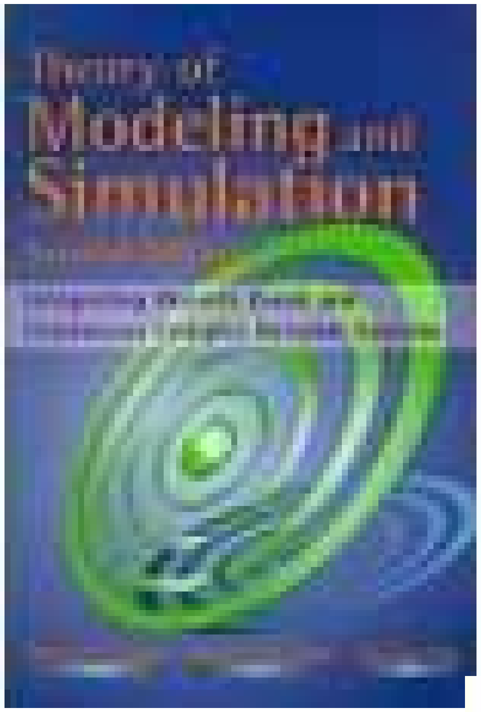
Model construction

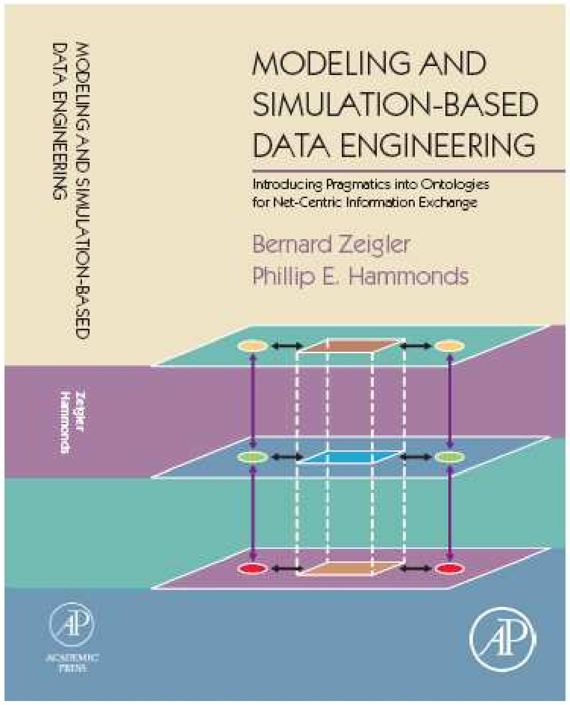


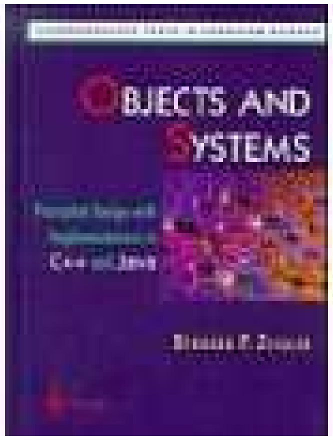
DEVS/SOA Infrastructure: Supports Deployment and Execution of DEVS Models on the Web


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

Books and Web Links







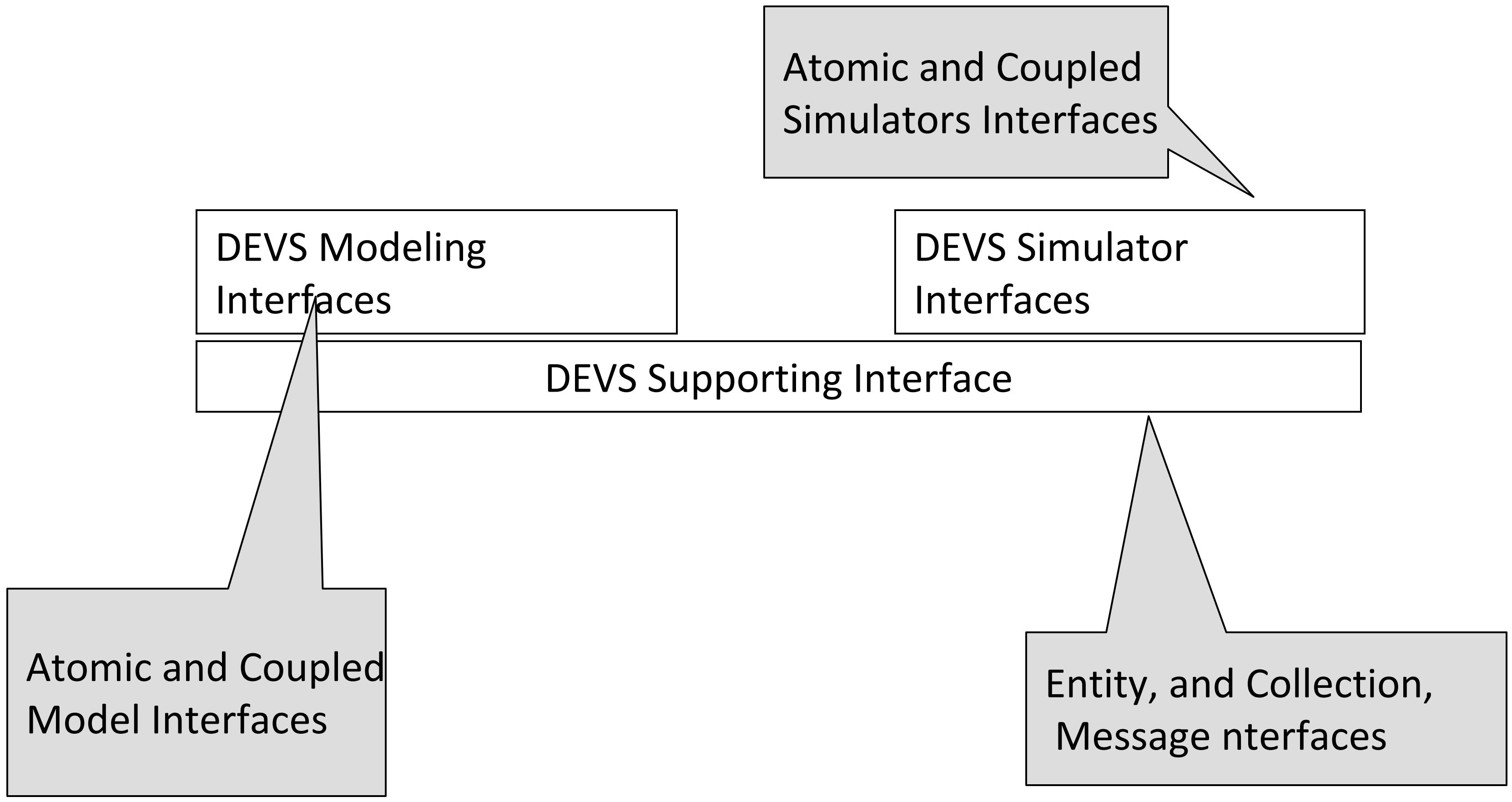
devsworld.org

acims.arizona.edu

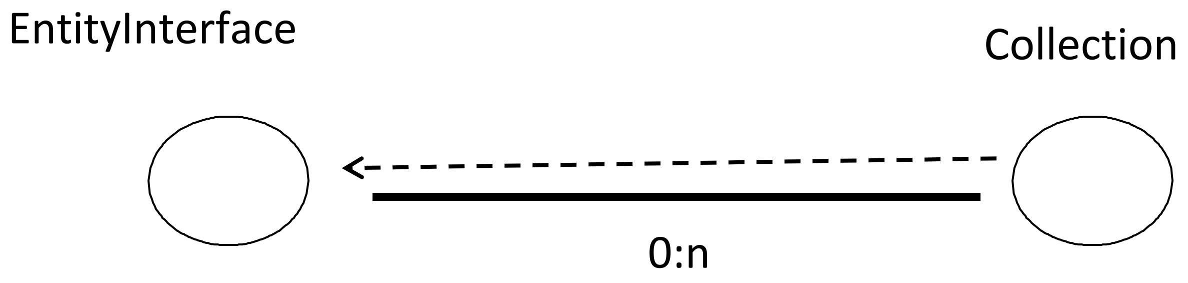
Rtsync.com

Backup: Proposed DEVS Standard

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

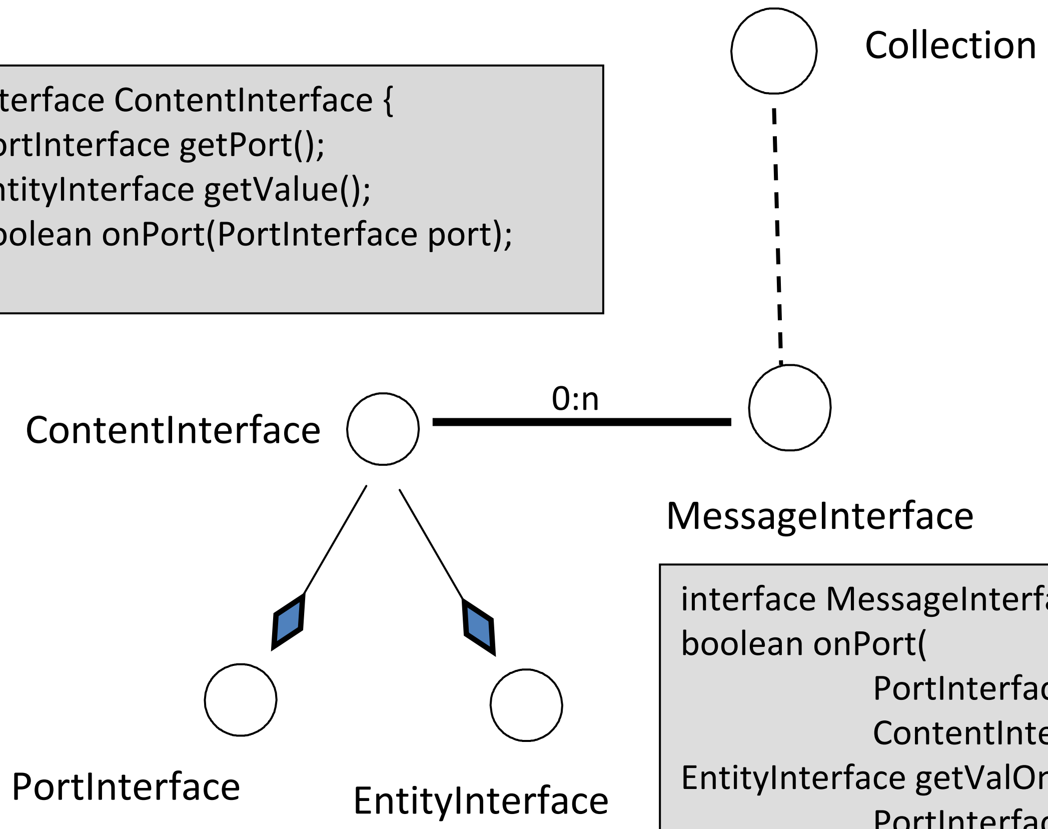


```
interface EntityInterface{
String getName();
boolean equalName(String name);
}
```

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

Message-related interfaces

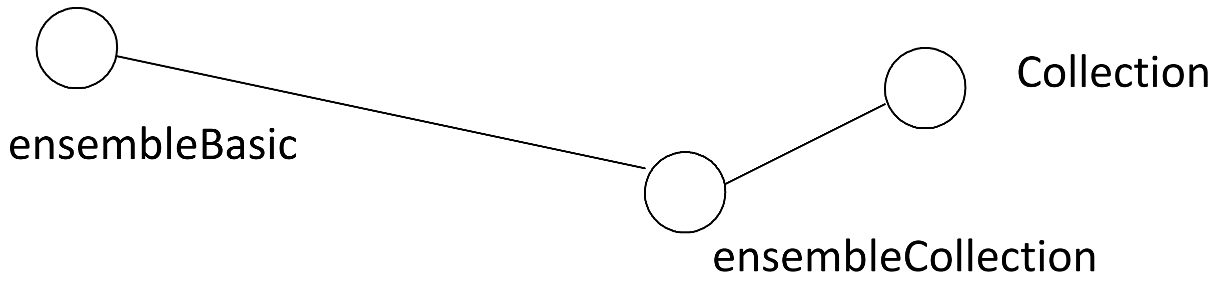
```
interface ContentInterface {
PortInterface getPort();
EntityInterface getValue();
boolean onPort(PortInterface port);
}
```



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

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

Ensemble Interfaces



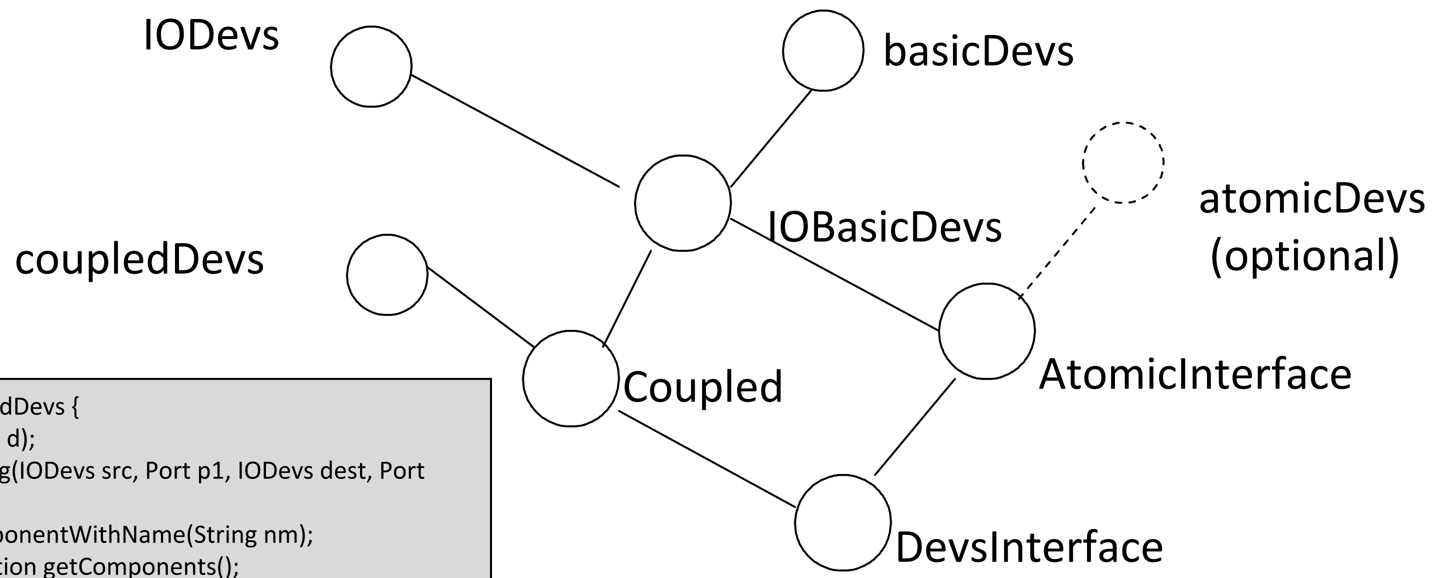
```
interface ensembleBasic {
void tellAll(Method m, EntityInterface[ ] args);
ensembleCollection askAll(Method m);
ensembleCollection which(Method m);
EntityInterface whichOne(Method m);
}

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

DEVS Model Interfaces

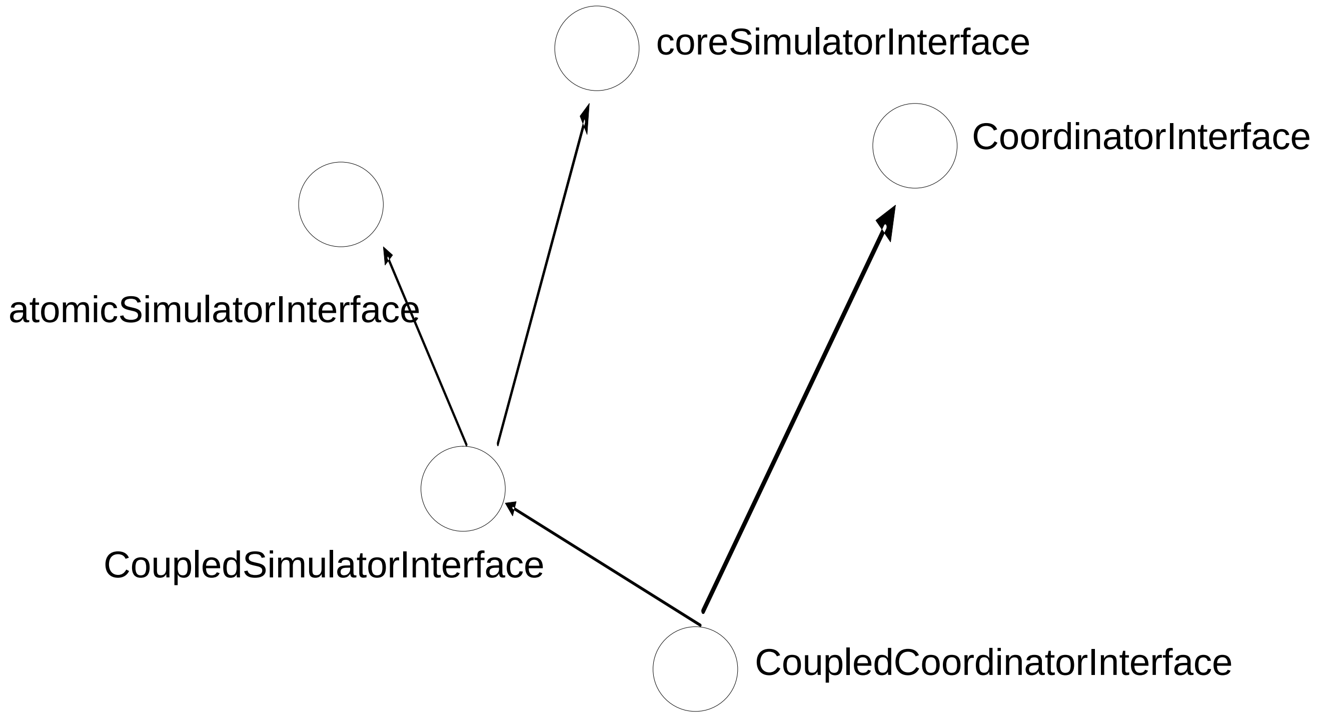
```
interface IODevs {
void addInport(String portName);
void addOutport(String portName);
ensembleCollection getInports();
ensembleCollection getOutports();
ContentInterface makeContent(PortInterface
port,EntityInterface value);
boolean   messageOnPort(MessageInterface x, PortInterface
port, ContentInterface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


coreSimulatorInterface
CoordinatorInterface
atomicSimulatorInterface
CoupledSimulatorInterface
CoupledCoordinatorInterface