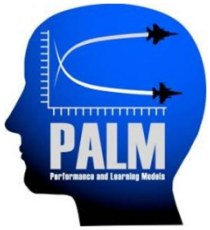


Net-centric Cognitive Architecture Using DEVS Unified Process

Saurabh Mittal, PhD

November 10, 2010

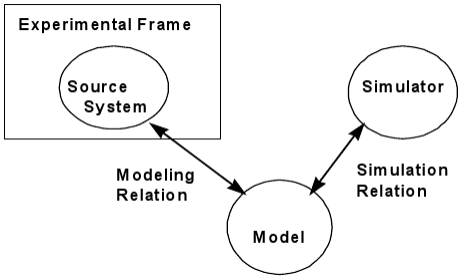
Outline

- DEVS Standard
- Model and Simulator Interoperability
- DEVS Unified Process

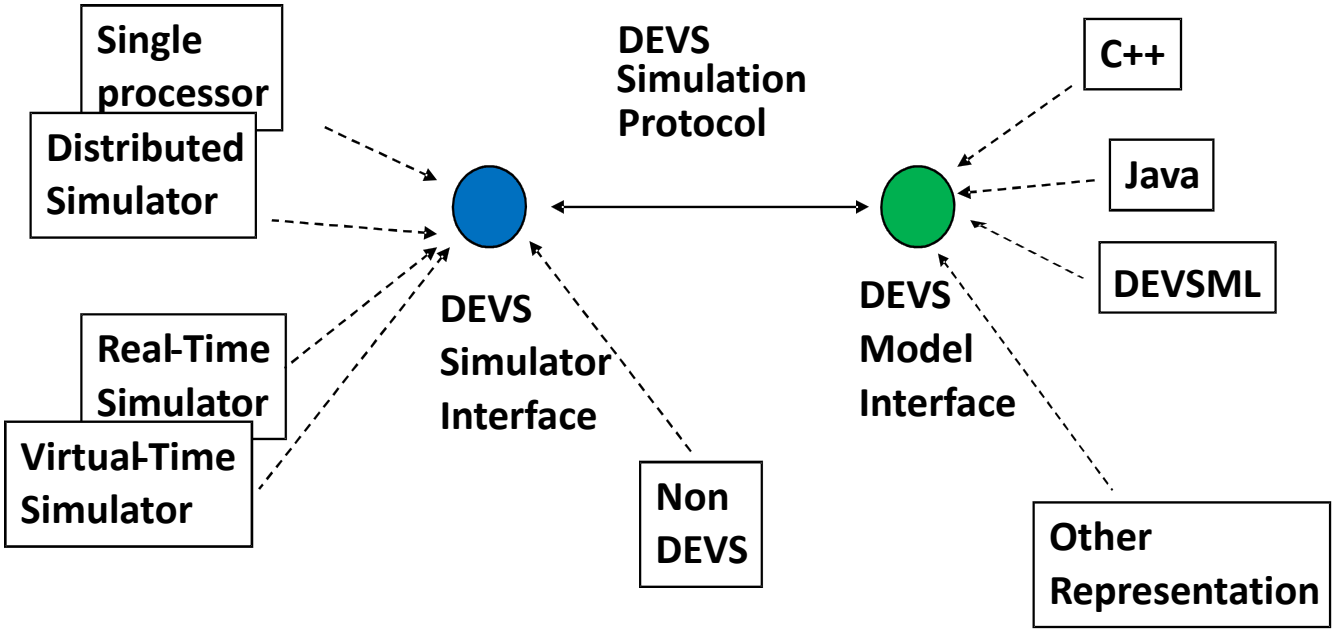
- Synthesis of Cognitive Agents
- Net-centric DEVS/ACT-R

- Cognitive System as a SoS component
- Summary

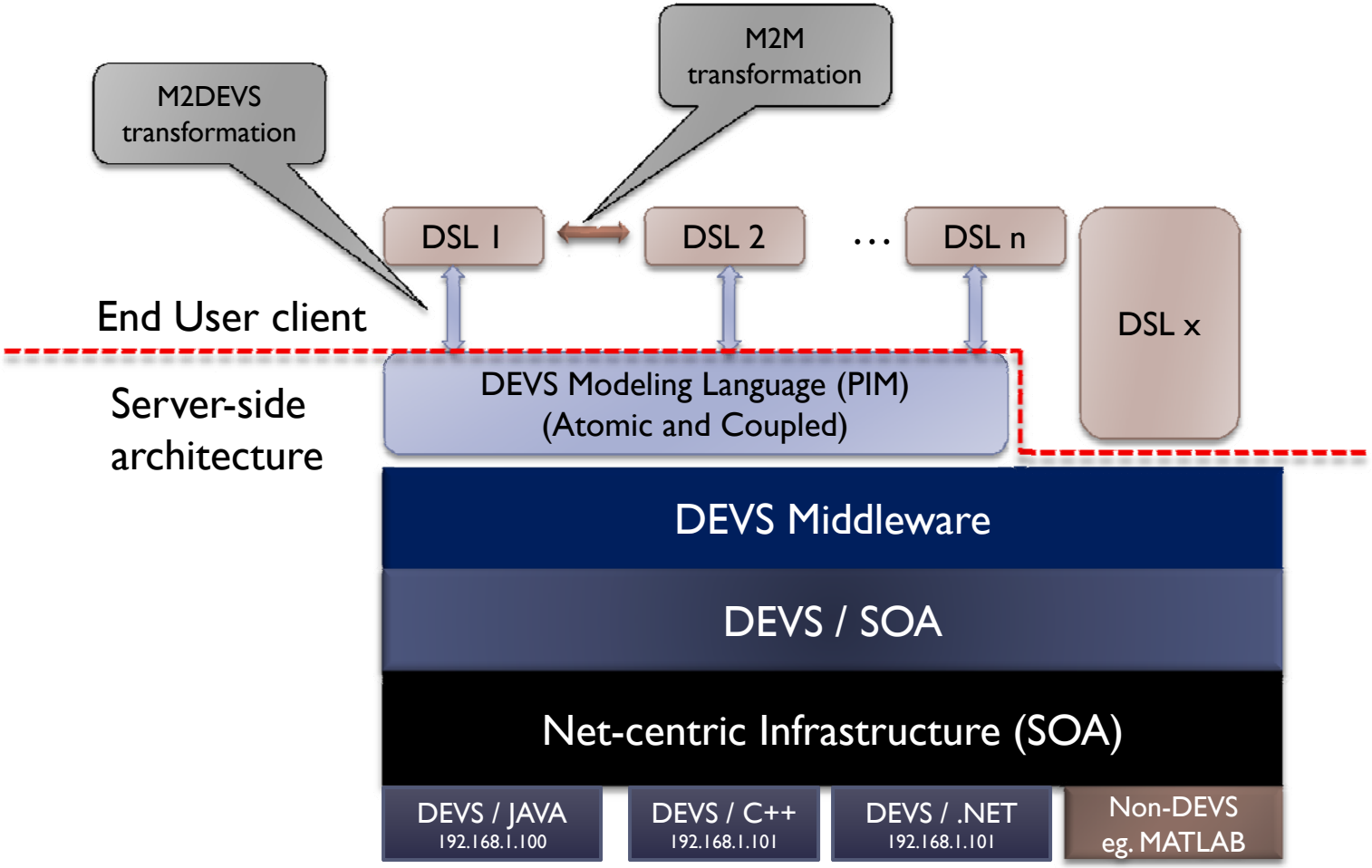
DEVS Standard



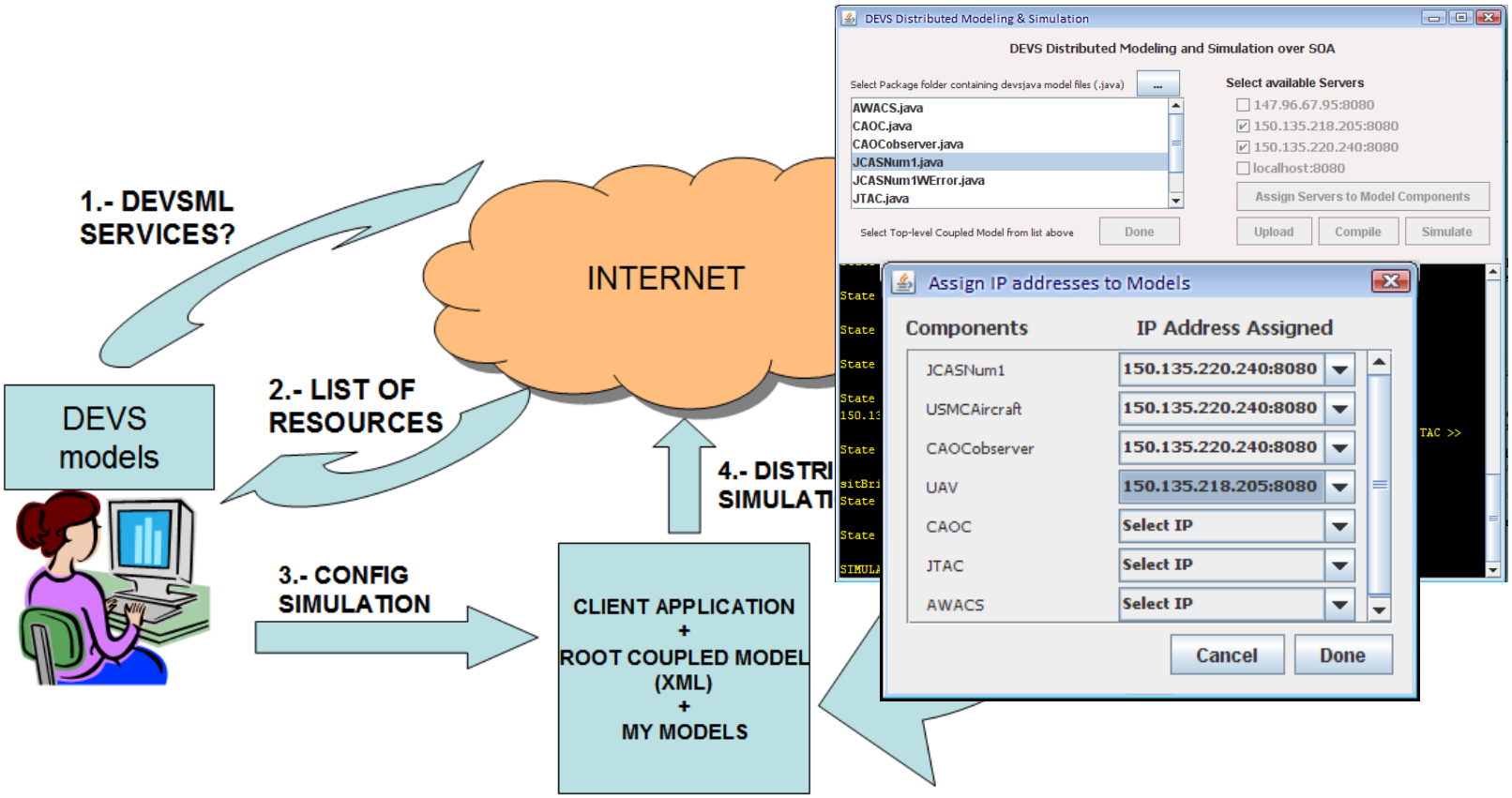
- Allows different modeling platforms and model portability, leading to model repositories
- Allows independent development of varied simulation engines



Model & Simulator Interoperability



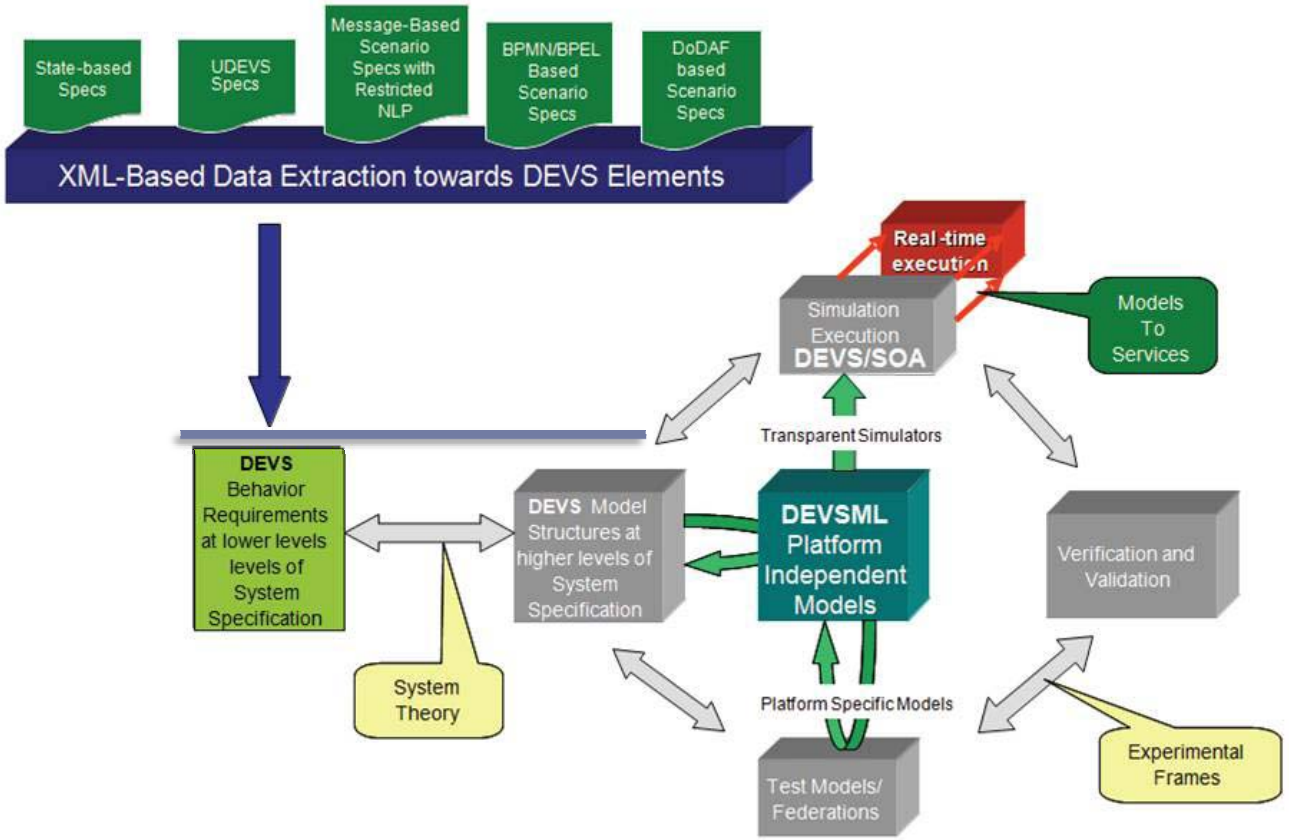
DEVS/SOA Operation



DEVS Unified Process

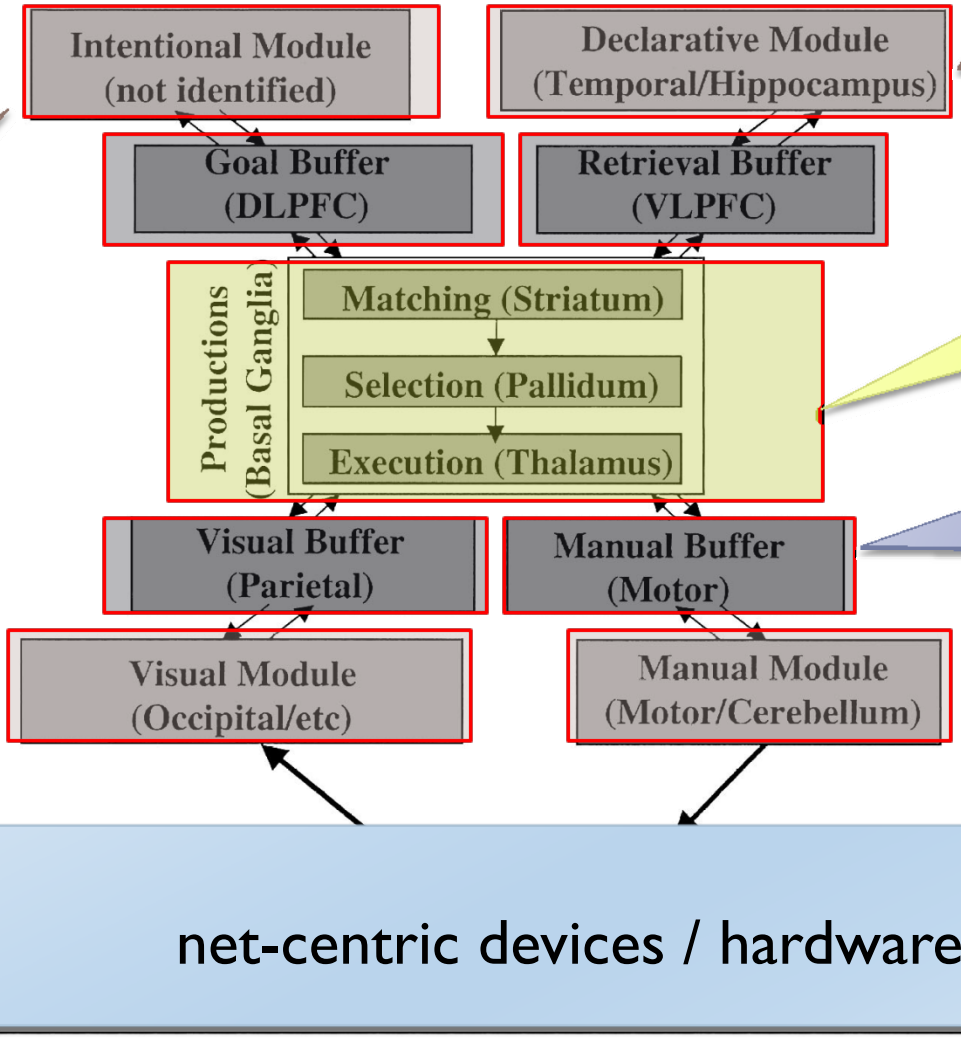
Level	Name	What we specify at this level
4	Coupled Systems	System built up by several component systems which are coupled together
3	I/O System	System with state and state transitions to generate the behavior
2	I/O Function	Collection of input/output pairs constituting the allowed behavior partitioned according to the initial state the system is in when the input is applied
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

- System Requirements as the starting point
- DEVS Hierarchical System specifications



Extracting Components from ACT-R

DEVS MODEL components



-scalable?
-production ready?
-network-ready?

Modules
-What's specific I/O
-do they have intrinsic behavior
-computational requirements?
-Hardware-in-the-loop?

Production
Network
-flat?
-Hierarchical?

Buffers
-do they respond differently to different Chunks?
-do they have intrinsic behavior?

net-centric devices / hardware

Formalizing DEVS/ACT-R

Production P= (D, B_p, Z, C, A)

where,

D is usual atomic DEVS $D=(X,S,Y,\delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta)$

B_p is set of Buffer proxies

Z is a set of bindings between buffer-slots and local variables

C is set of Conditions

A is set of Actions

P will be in DSL / XML, with DEVS *D* in PSM as *an 'ACT-R primitive'*

Some Other ACT-R primitives:

- Slot SL = (Key,Value)
- ChunkType CT = <SL>
- Chunk CH = (CT_{type}, <SL>)
- SlotCompare SLC= (SL, $BOOL_{equals}$)
- Condition C= (P_{name}, B_{name}, $BOOL_{clearBuf}$, <SLC>)
- Action A = (P_{name}, B_{name}, CT, $BOOL_{clearBuf}$, <SL>)

- **Buffer B** = (D,CH,$DOUBLE_{procesingTime}$)
- **Module Declarative Mem MDM** = (D, <CH>)
- **Selector S** = (D)
- **ActrModel M** = (C, <CT>,<P>)

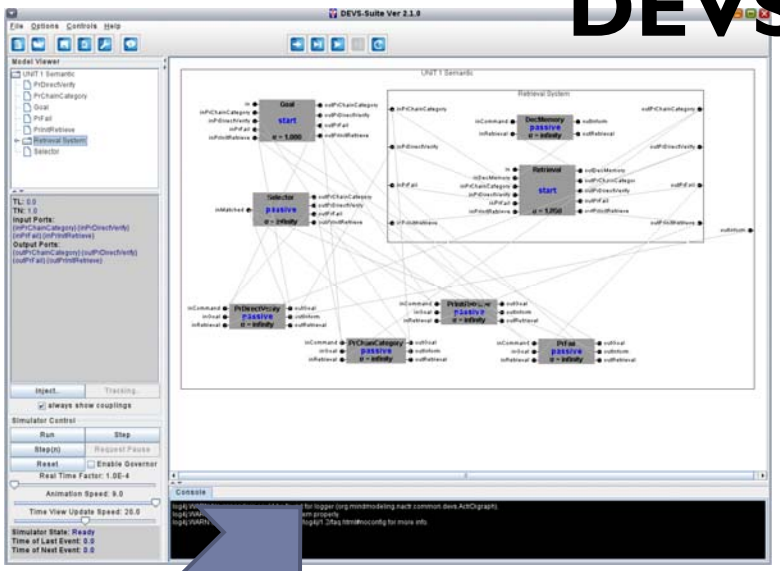
  where C is coupled DEVS C = (X,Y, M, EIC, EOC, IC)

Legend:
Atomic, Coupled
<m> = Set of m
(m) = single value

Casting in formal DEVS

```
production PrStart{
    production PrIncrement{
        binding =goal countfr
                count to num1
            }
        binding  =retrieval countorder{
            second to num2
        }

        condition =goal>ISA countfrom{
            end != num1
        }
        condition =retrieval>ISA countorder{
            first == num1
        }
        action =goal>{
            count = num2
        }
    }
    action +retrieval>ISA count
        first = num2
    }
}
```

DSL / PIM

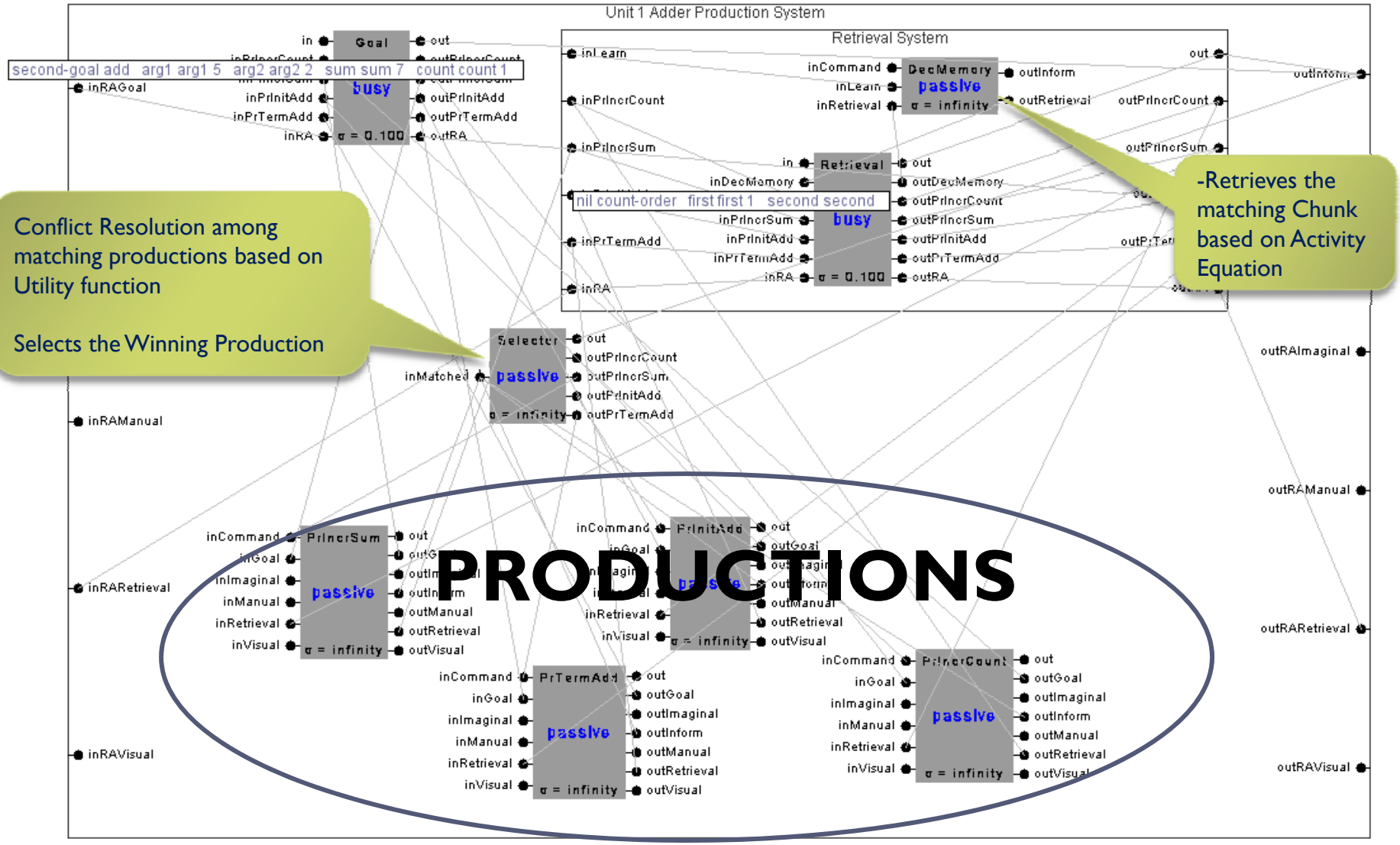


DEVS

PSM

```
@Override
public void gatherBufferVariables() {
    addBinding("value", gatherBufferSlot(CONST.VISUAL, "value"));
}

@Override
public void loadConditions() {
    Condition c = new Condition("evalGoal", getName(), CONST.GOAL, "read-letters");
    addSlotCondition(c, "state", "attend", true);

    Condition c2 = new Condition("evalVisual", getName(), CONST.VISUAL, "text");
    addCondition(c2);
}

@Override
public void loadActions() {
    Action a = new Action("setGoal", getName(),CONST.GOAL);
    addUpdateSlotAction(a, "state", "respond");

    Action a2 = new Action("setImaginal", getName(),CONST.IMAGINAL, true, getChunkType("array"));
    addUpdateSlotAction(a2, "letter", getBinding("value"));
}
```

Casting in formal DEVS



Log Trace Matches ACT-R

time

Set Goal

Conflict Resolution

Select Winner

Fire Production

Set Goal

Request Retrieval

Retrieved Chunk x

Set Buffer x

```
0.000   GOAL                    SET-BUFFER-CHUNK GOAL FIRST-GOAL REQUESTED NIL
0.000   PROCEDURAL              CONFLICT-RESOLUTION
0.000   PROCEDURAL              PRODUCTION-SELECTED START
0.000   PROCEDURAL              BUFFER-READ-ACTION GOAL
0.050   PROCEDURAL              PRODUCTION-FIRED START
0.050   PROCEDURAL              MOD-BUFFER-CHUNK GOAL
0.050   PROCEDURAL              MODULE-REQUEST RETRIEVAL
0.050   PROCEDURAL              CLEAR-BUFFER RETRIEVAL
0.050   DECLARATIVE             START-RETRIEVAL
0.050   PROCEDURAL              CONFLICT-RESOLUTION
0.100   DECLARATIVE             RETRIEVED-CHUNK C
0.100   DECLARATIVE             SET-BUFFER-CHUNK RETRIEVAL C
0.100   PROCEDURAL              CONFLICT-RESOLUTION
0.100   PROCEDURAL              PRODUCTION-SELECTED INCREMENT
0.100   PROCEDURAL              BUFFER-READ-ACTION GOAL
0.100   PROCEDURAL              BUFFER-READ-ACTION RETRIEVAL
0.150   PROCEDURAL              PRODUCTION-FIRED INCREMENT
```

ACT-R Trace

DEVS/ACT-R Trace

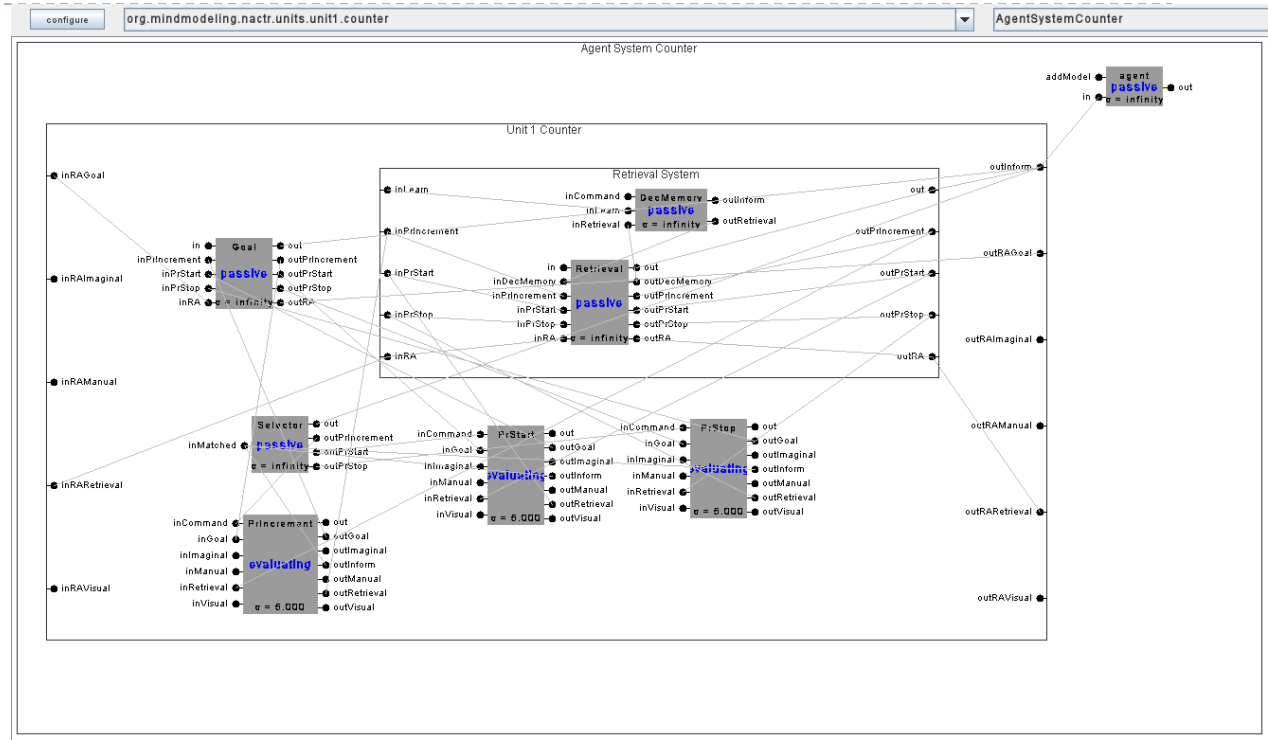
```
[09:47:50,363] INFO  - [UNIT 1 Semantic] BEGINING...
[09:47:50,363] INFO  - [UNIT 1 Semantic] defining Declarative Memory chunks...
[09:47:50,365] INFO  - [Goal] SET BUFFER: [chunk:{g3,{{object,canary} {category,fish} {judgement,null} }]
[09:47:51,900] INFO  - [Goal] REFRESHING PRODUCTIONS
[09:47:51,913] INFO  - [PrInitRetrieve] MATCHED
[09:47:51,915] INFO  - [Selector]  CONFLICT RESOLUTION
[09:47:51,916] INFO  - [Selector]  SELECTED Production PrInitRetrieve
[09:47:51,919] INFO  - [PrInitRetrieve] FIRED
[09:47:51,924] INFO  - [Goal] SET BUFFER: [chunk:{g3,{{object,canary} {category,fish} {judgement,pending} }]
[09:47:51,924] INFO  - [Retrieval] REQUEST RETRIEVAL
[09:47:51,926] INFO  - [Retrieval] CHUNK TYPE TO RETRIEVE: property
[09:47:51,928] INFO  - [Goal] REFRESHING PRODUCTIONS
[09:47:51,941] INFO  - [DecMemory] START RETRIEVAL
[09:47:51,941] INFO  - [DecMemory] COMPLETED SEARCH
[09:47:51,942] INFO  - [DecMemory] RETRIEVED: [chunk:{p14,{{object,canary} {attribute,category} {value,bird} }]
[09:47:51,945] INFO  - [Retrieval] SET BUFFER: [chunk:{p14,{{object,canary} {attribute,category} {value,bird} }]
[09:47:51,947] INFO  - [Retrieval] REFRESHING PRODUCTIONS
[09:47:51,959] INFO  - [PrChainCategory] MATCHED
[09:47:51,962] INFO  - [Selector]  CONFLICT RESOLUTION
[09:47:51,962] INFO  - [Selector]  SELECTED Production PrChainCategory
[09:47:51,965] INFO  - [PrChainCategory] FIRED
[09:47:51,970] INFO  - [Goal] SET BUFFER: [chunk:{g3,{{object,bird} {category,fish} {judgement,pending} }]
```

Cognitive Agents

- Motivation:
  - Reuse existing ACT-R models
  - Production networks are flat and not scalable but we need the aggregate behavior of model
- Production is a rule-action pair. So, is the entire ACT-R model ~ A cognitive capacity?
  - that executes an 'action' sequence based on certain 'conditions'
- Can we compress an ACT-R production network to a single atomic and not loose behavior?
- Solution provided by DEVS closure under coupling that allows us to treat a coupled model an atomic one.
- OR should we start with baseline cognitive capacities and build the Agent's behavior repertoire?

Cognitive Agent Synthesis

- Agent observes
  - Goal Updates
  - Retrieval Requests
  - Retrieval Outcomes
  - Winner Productions
- Relevant Input Stream

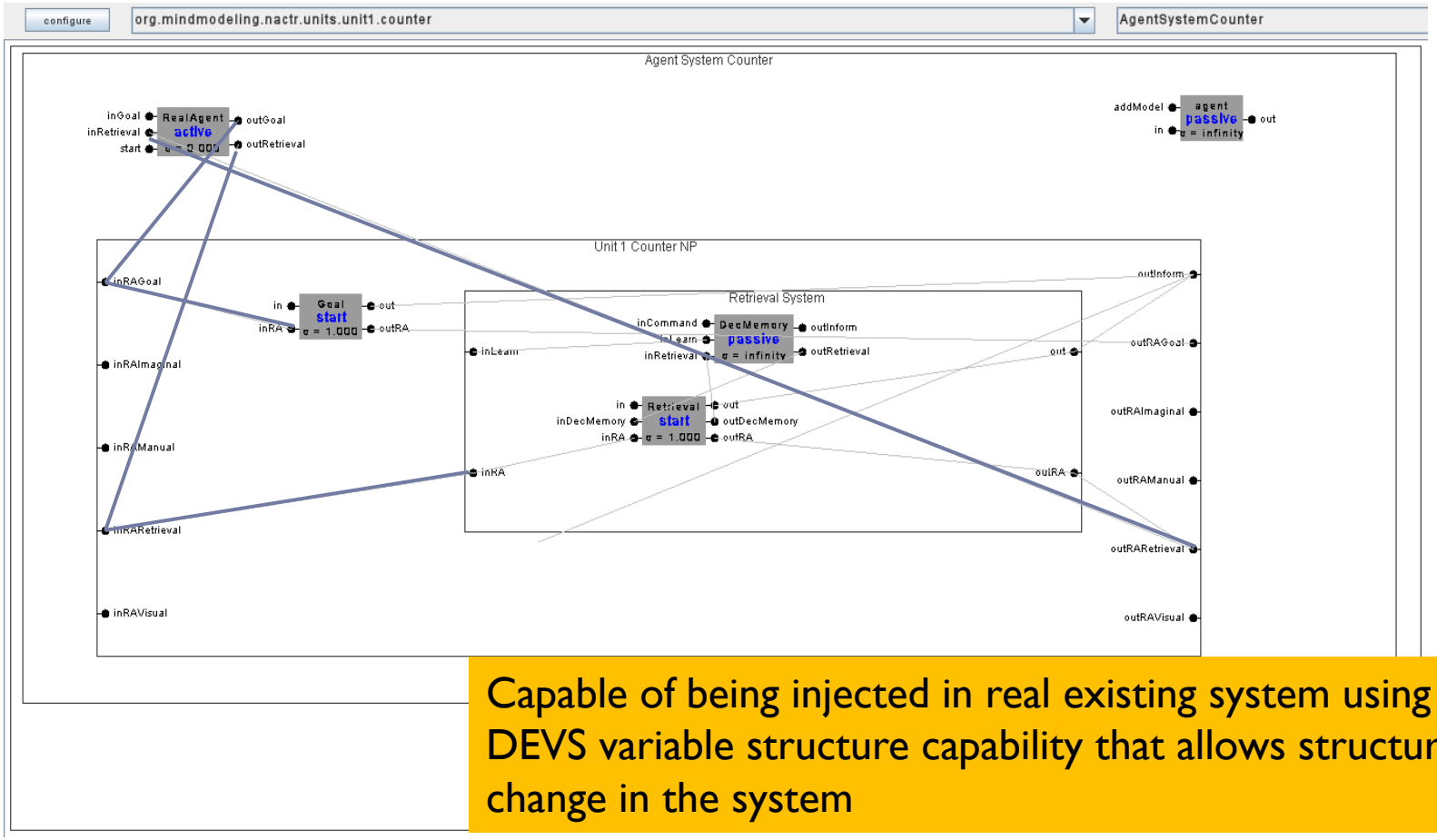


Events are used to extract:

1. Time-ordered Goals
2. Time-ordered Retrievals
3. Time-ordered Expectations

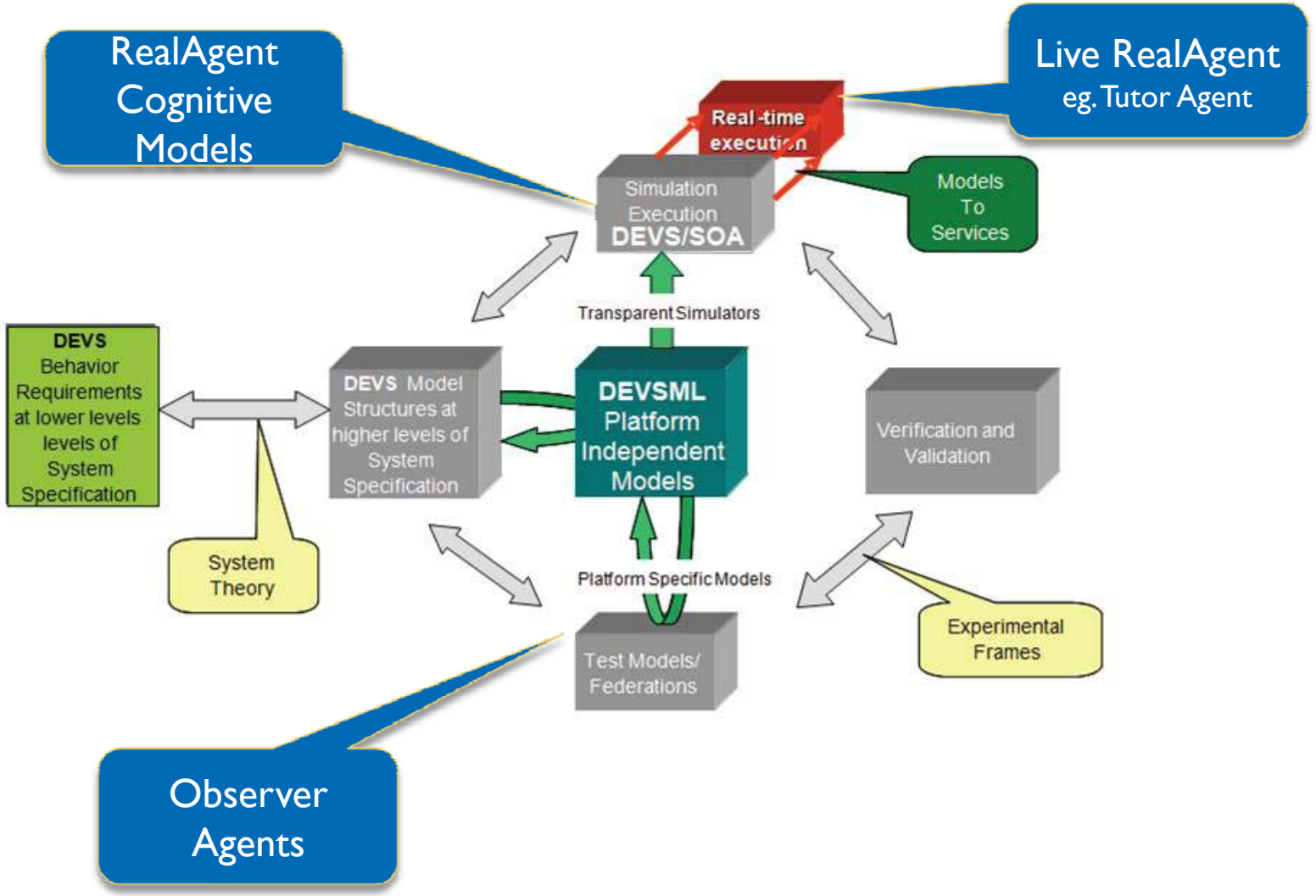
Cognitive Agent Synthesis

- Same behavior and no need of production network
- Develop multi-agent network with a coordinator/manager to manage the behavior repertoire

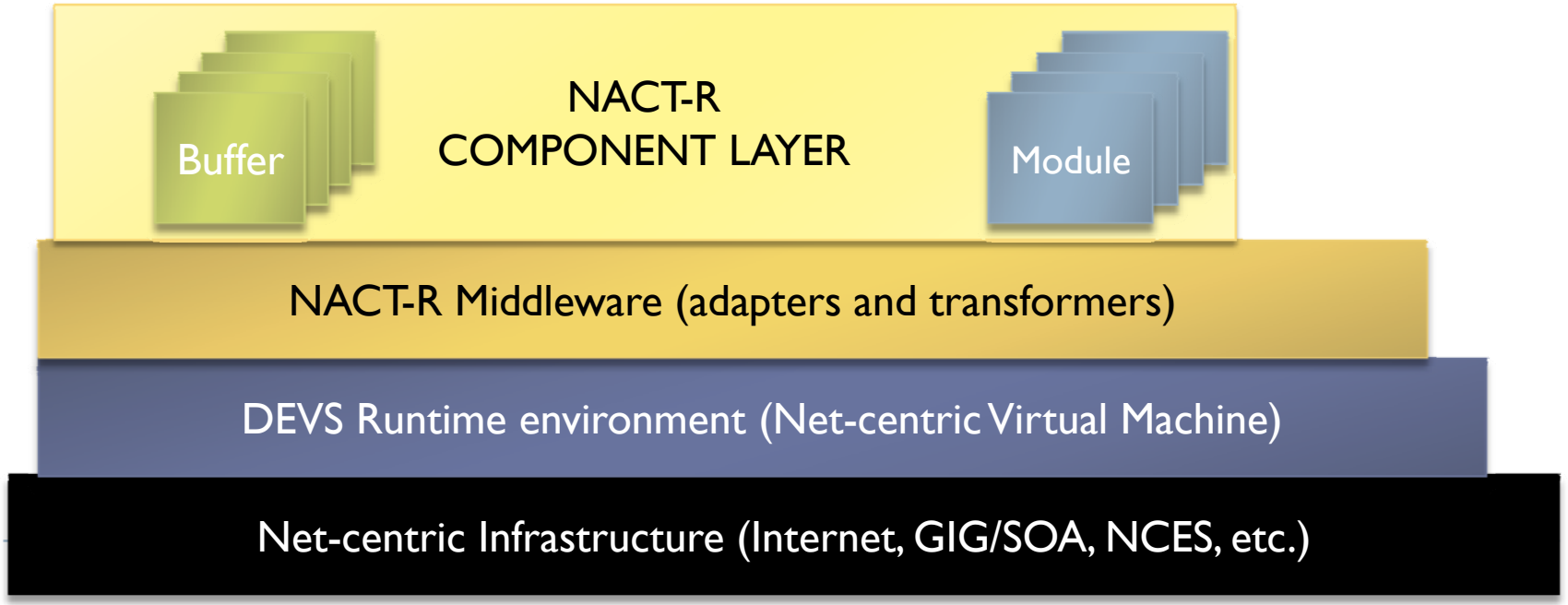


Capable of being injected in real existing system using DEVS variable structure capability that allows structural change in the system

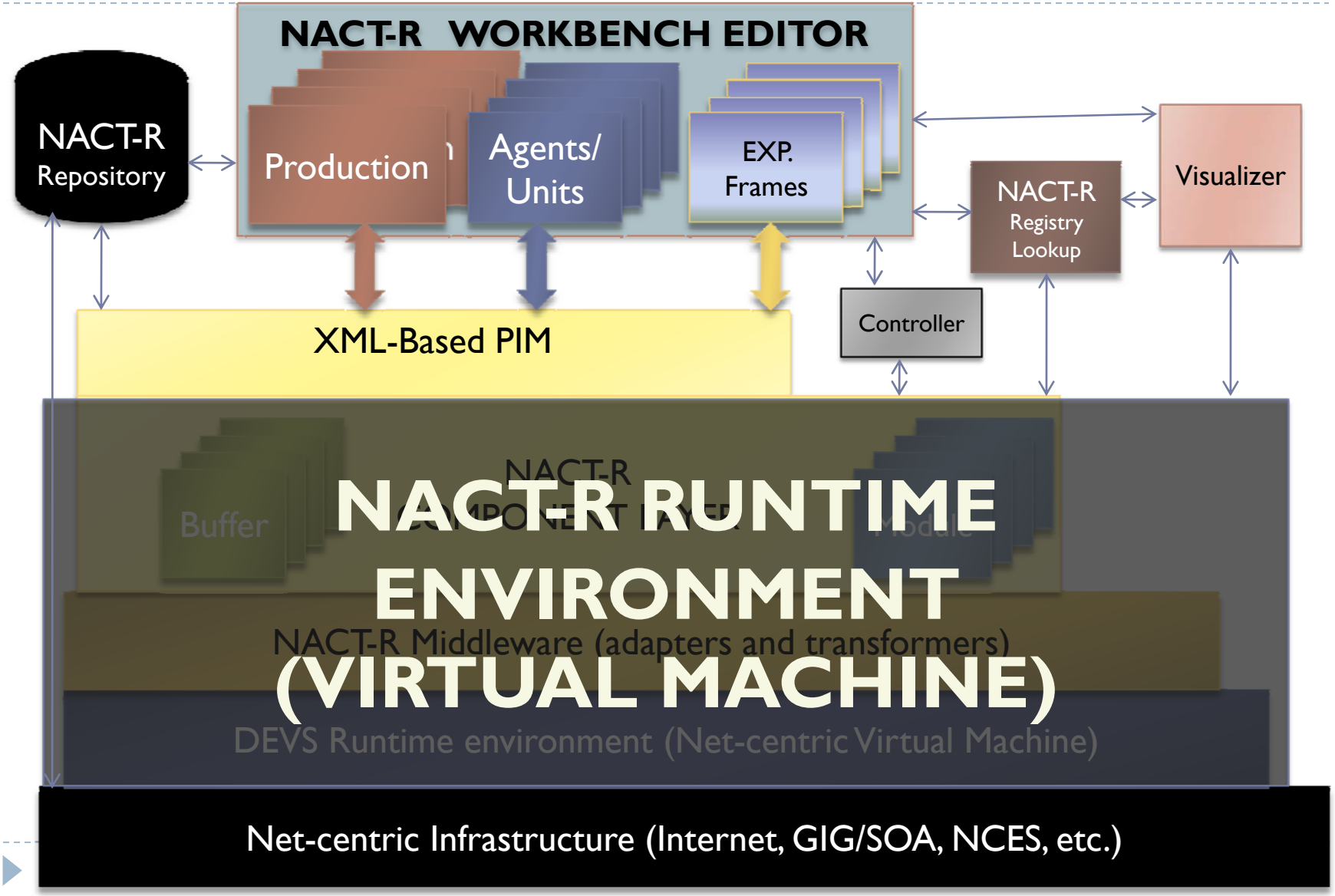
Recalling DEVS Unified Process



DEVS/Net-centric ACT-R Architecture



DEVS/Net-centric ACT-R Architecture



DEVS/Net-centric ACT-R Architecture

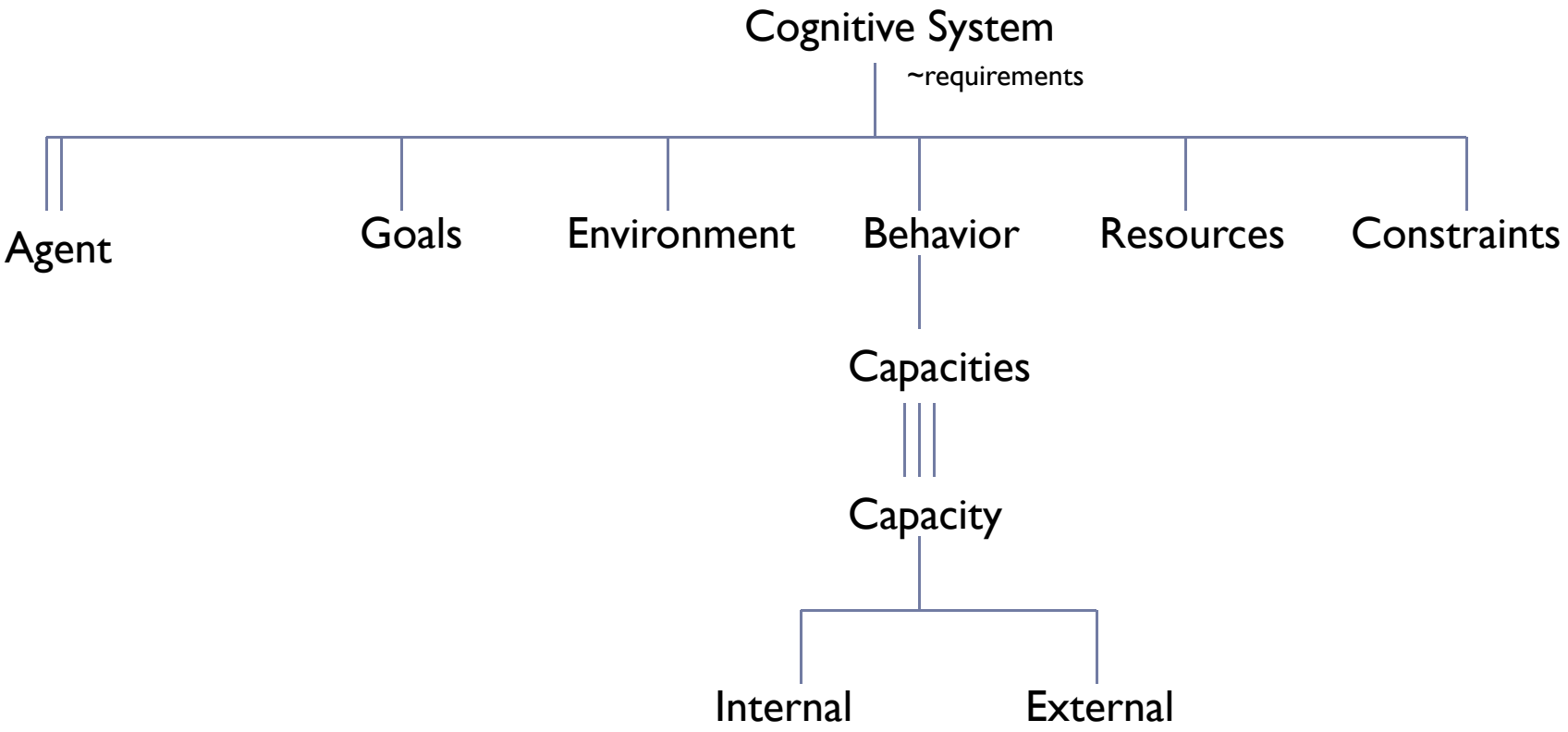
Cognitive System as a Component

ACT-R can be component based using DEVS but is still not reusable due to the resolution at the Production level. We need capacities!

Extending ACT-R:

1. Extract aggregate behavior from ACT-R models in cognitive agents using DEVS hierarchy of system specifications
2. Develop a repository of such cognitive capacities that can be coupled together towards a Behavior Repertoire
3. Observers can be used to evaluate 'equivalence' between two similar ACT-R models
4. Include Top level Goals, Resources and Situatedness per 'System' requirements or Mission Thread requirements
5. Compose agent's 'capabilities' *insitu* with higher level goals, constraints and environment specified/managed using Experimental Frames
6. Exploit dynamism in Environment, Resources, Behavior and Goals with DEVS variable structure.
7. Keep it modular i.e. promote interfaces of constituent components.

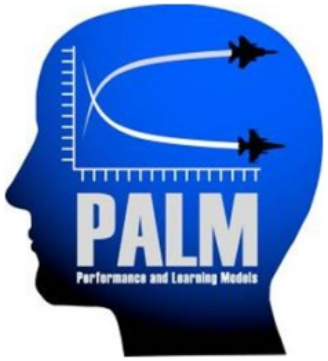
Agent as a 'System'



Summary

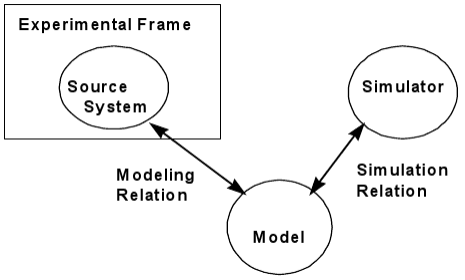
- DEVS is component based framework with PIM and transparent simulators in net-centric domain
- DEVS Unified Process and SES allows multi-formalism M&S
- ACT-R can be made extended with well-defined component interfaces
- DEVS hierarchy of System specifications allows development of components by observing at lower levels of specification
- Cognitive Agents or capacities can be created from existing repository of ACT-R models
- Cognitive agents can now reap the benefits of integration with larger System of Systems or 'human operator'
- DEVS agent models become live entities that can interface with live web services or hardware *(the model-continuity principle)*.
- DEVS is a production system with near zero transition cost from Model to deployable Software.

Questions & Comments

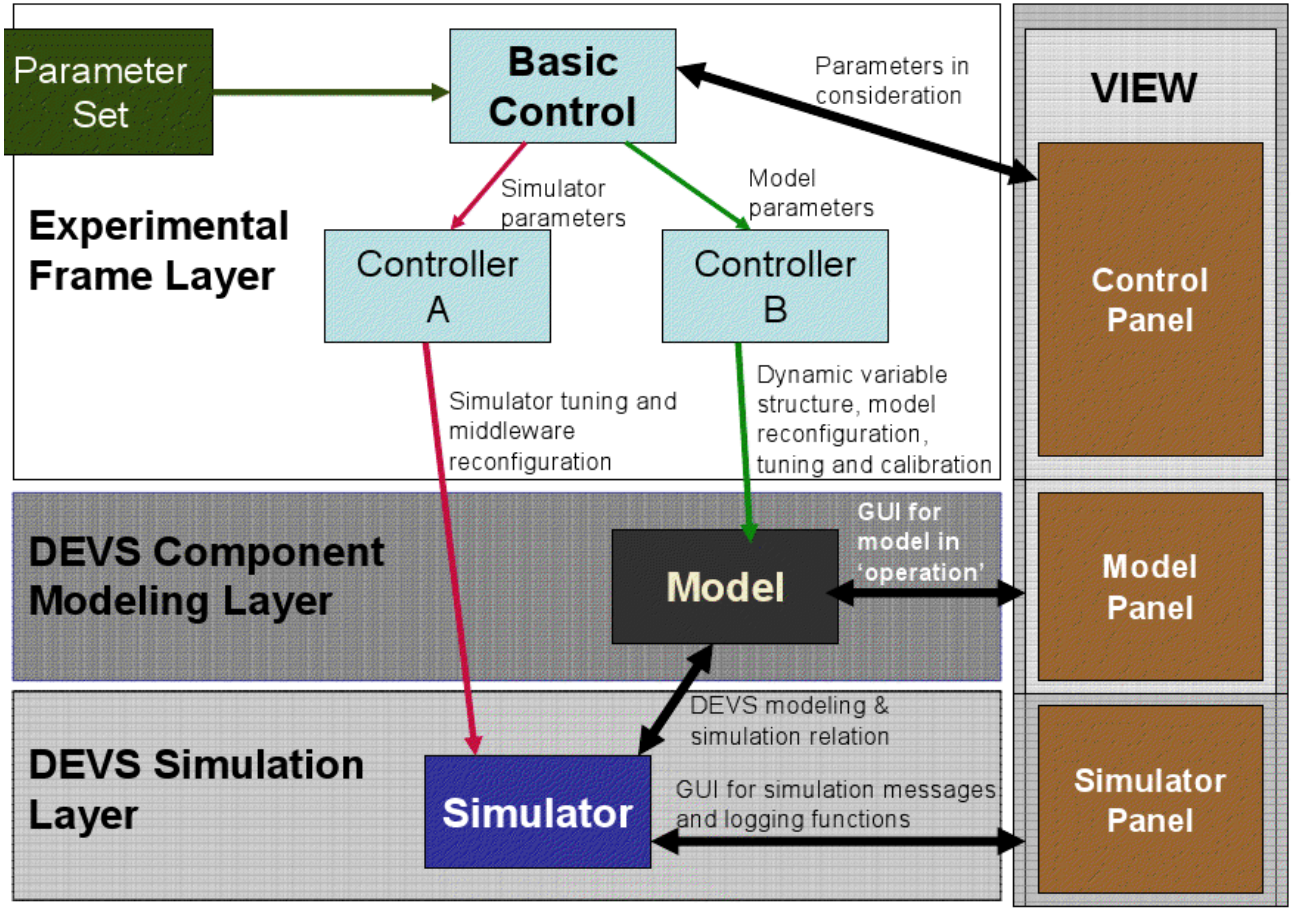


Backup Slides

DEVS MSVC Paradigm



- Model-Simulator-View-Controller

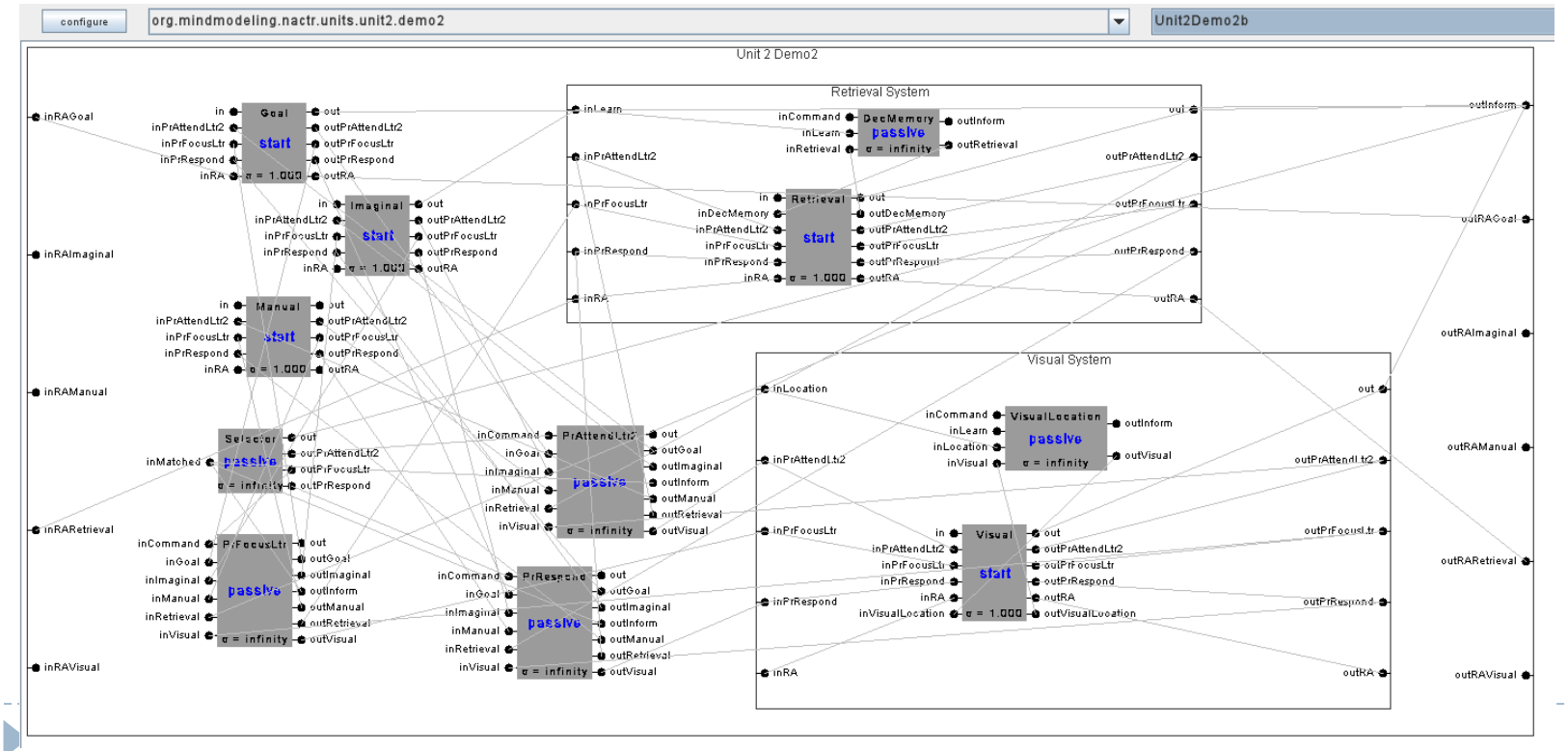


DEVS Hierarchy of System Specification

Level	Name	What we specify at this level
4	Coupled Systems	System built up by several component systems which are coupled together
3	I/O System	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 the initial state the system is in when the input is applied
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/NACT-R Unit 2

- Visual Buffer and Visual-Location coupled together
  - Act of 'looking' (move-attention + encoding)
  - Reducing the number of productions



Unit 2: Demo 1

- Attend Letter

```
@Override
public void loadConditions() {
    Condition c = new Condition("evalGoal", getName(), CONST.GOAL, "read-letters");
    addSlotCondition(c, "state", "start", true);
}

@Override
public void loadActions() {
    Action a = new Action("setGoal", getName(),CONST.GOAL);
    addUpdateSlotAction(a, "state", "attend");

    Action a2 = new Action("setVisualLocation", getName(), CONST.VISUAL, true, getChunkType("visual-location"));
    addUpdateSlotAction(a2, "attended", "nil");
}
```

Action Requesting Visual System for a visual location (move attention)

- Focus Letter

```
@Override
public void gatherBufferVariables() {
    addBinding("value", gatherBufferSlot(CONST.VISUAL, "value"));
}

@Override
public void loadConditions() {
    Condition c = new Condition("evalGoal", getName(), CONST.GOAL, "re...ters");
    addSlotCondition(c, "state", "attend", true);

    Condition c2 = new Condition("evalVisual", getName(), CONST.VISUAL, "te...
    addCondition(c2);
}

@Override
public void loadActions() {
    Action a = new Action("setGoal", getName(),CONST.GOAL);
    addUpdateSlotAction(a, "state", "respond");

    Action a2 = new Action("setImaginal", getName(),CONST.IMAGINAL, true, getChunkType("array"));
    addUpdateSlotAction(a2, "letter", getBinding("value"));
}
```

Checking condition that Visual system has completed the 'encoding' process and is filled with 'Object'

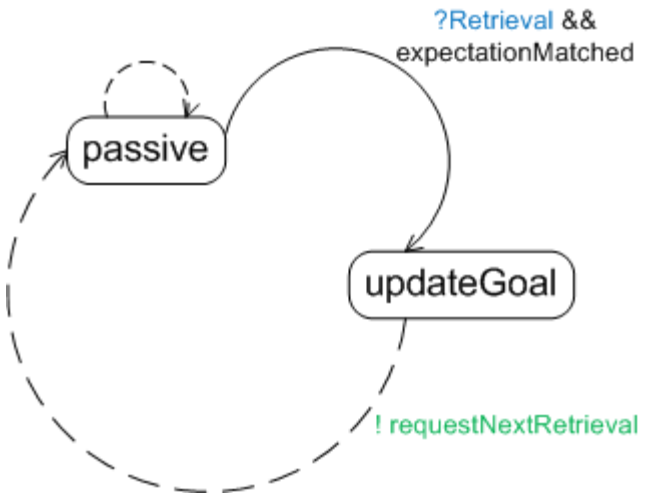
Action to put in 'Imaginal'

Letter

```
addBinding("letter", gatherBufferSlot(CONST.IMAGINAL, "letter"));
}

@Override
public void loadConditions() {
    Condition c = new Condition("evalGoal", getName(), CONST.GOAL, "read-letters");
    addSlotCondition(c, "state", "respond", true);

    dition c2 = new Condition("evalImaginal", getName(), CONST.IMAGINAL, "array");
    Condition(c2);


ide
public void loadActions() {
    Action a = new Action("setGoal", getName(),CONST.GOAL);
    addUpdateSlotAction(a, "state", "done");

    Action a2 = new Action("setManual", getName(), CONST.MANUAL, true, getChunkType("press-key"));
    addUpdateSlotAction(a2, "key", getBinding("letter"));
}
```

Real Agent Behavior

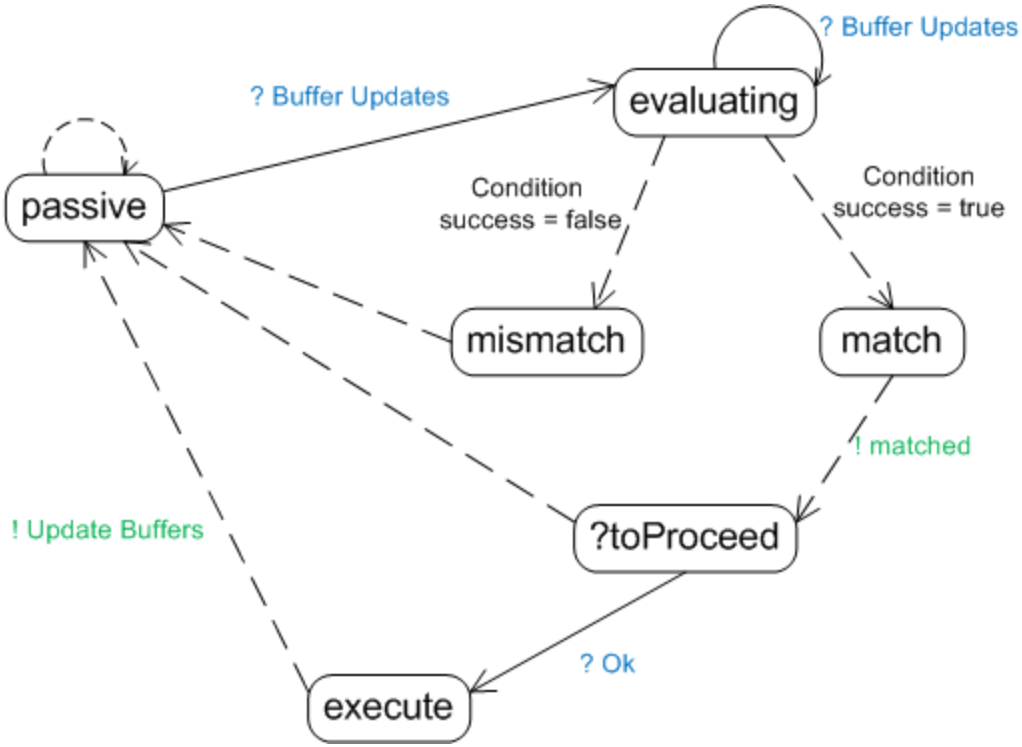
- Set of Time ordered Goals
- Set of Time ordered Retrievals
- Set of Time ordered Expectations



1. Setting a new goal is linked to requesting the next retrieval
2. Once the expected retrieval is success, a new goal is set

Production primitive

```
production PrIncrement{
    binding =goal countfrom (
            count to num1
        )
    binding  =retrieval countorder(
        second to num2
    )

    condition =goal>ISA countfrom(
        end != num1
    )
    condition =retrieval>ISA countorder(
        first == num1
    )
    action =goal>(
        count = num2
    )
    action +retrieval>ISA countorder(
        first = num2
    )
}
```



Cognitive Agent Synthesis - 4

Log generated by RealAgent

```
[10:44:15,331] INFO     - [RealAgent] Total goals: 4
[10:44:15,331] INFO     - [RealAgent] Total retrievals: 3
[10:44:15,331] INFO     - [RealAgent] Total bufferUpdates: 3
[10:44:15,336] DEBUG    - [Unit 1 Counter NP] Count-order defined
[10:44:15,336] DEBUG    - [Unit 1 Counter NP] Count-from defined
[10:44:15,337] INFO     - [Unit 1 Counter NP] ChunkTypes defined.
[10:44:15,337] INFO     - [Unit 1 Counter NP] defined chunk types
[10:44:15,337] INFO     - [Unit 1 Counter NP]


================================================================================
[10:44:15,337] INFO     - [Unit 1 Counter NP] BEGINING...
[10:44:15,337] INFO     - [Unit 1 Counter NP] defining Declarative Memory chunks...
[10:46:00,502] INFO     - [RealAgent] Setting goal: <Chunk>
  <name>first-goal</name>
  <chunkType>count-from</chunkType>
  <slots>
    <slot>
      <name>start</name>
      <key>start</key>
      <value>2</value>
    </slot>
    <slot>
      <name>end</name>
      <key>end</key>
      <value>4</value>
    </slot>
    <slot>
      <name>count</name>
      <key>count</key>
    </slot>
  </slots>
</Chunk>
[10:46:02,005] INFO     - [RealAgent] Requesting retrieval: <Chunk>
  <name>nil</name>
  <chunkType>count-order</chunkType>
  <slots>
    <slot>
      <name>first</name>
      <key>first</key>
      <value>2</value>
    </slot>
    <slot>
      <name>second</name>
      <key>second</key>
    </slot>
  </slots>
</Chunk>
[10:46:02,014] INFO     - [Retrieval] REQUEST RETRIEVAL
[10:46:02,016] INFO     - [Retrieval] CHUNK TYPE TO RETRIEVE: count-order
[10:46:02,018] INFO     - [Goal] SET BUFFER: [chunk:{count-from,{{start,2} {end,4} {count,2} }]
[10:46:02,634] INFO     - [DecMemory] START RETRIEVAL
[10:46:02,634] INFO     - [DecMemory] COMPLETED SEARCH
[10:46:03,236] INFO     - [DecMemory] RETRIEVED: [chunk:{count-order,{{first,2} {second,3} }]
[10:46:03,242] INFO     - [Retrieval] SET BUFFER: [chunk:{count-order,{{first,2} {second,3} }]
[10:46:03,447] INFO     - [RealAgent] Retrieval Completed
[10:46:03,448] INFO     - [RealAgent] Expectation Matched
[10:46:03,448] INFO     - [RealAgent] Setting goal: <Chunk>
  <name>first-goal</name>
  <chunkType>count-from</chunkType>
  <slots>
    <slot>
      <name>start</name>
      <key>start</key>
      <value>2</value>
    </slot>
    <slot>
      <name>end</name>
      <key>end</key>
      <value>4</value>
    </slot>
    <slot>
      <name>count</name>
      <key>count</key>
      <value>2</value>
    </slot>
  </slots>
</Chunk>
[10:46:03,648] INFO     - [RealAgent] Requesting retrieval: <Chunk>
  <name>nil</name>
  <chunkType>count-order</chunkType>
```

DEVS/Net-centric ACT-R Architecture