



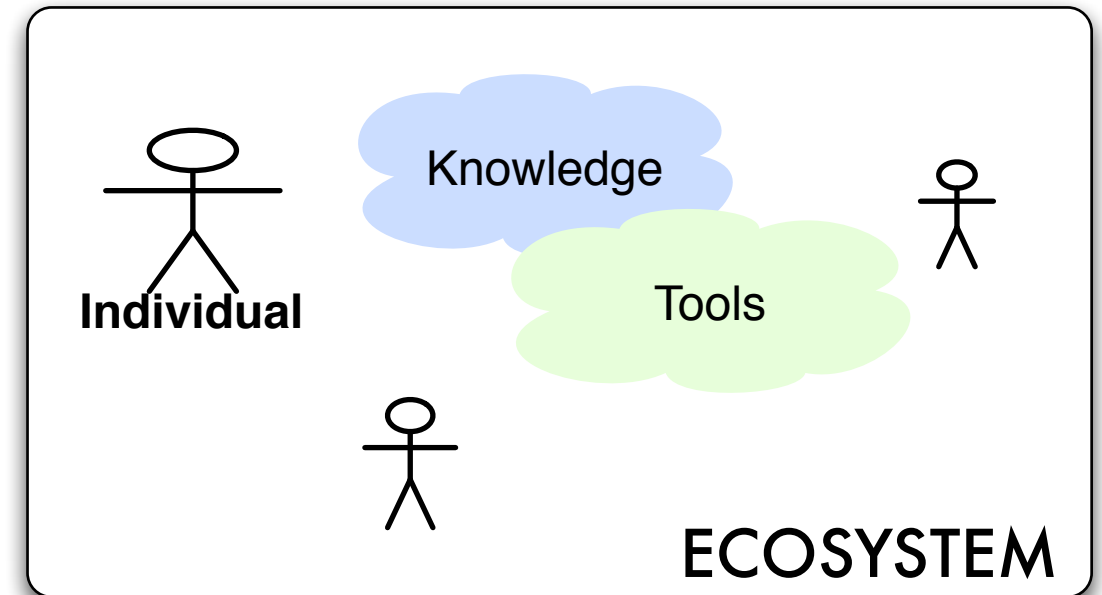
# Goal Directed Agents using Artifacts

**Alessandro Ricci and Michele Piunti**

[michele.piunti@unibo.it](mailto:michele.piunti@unibo.it)

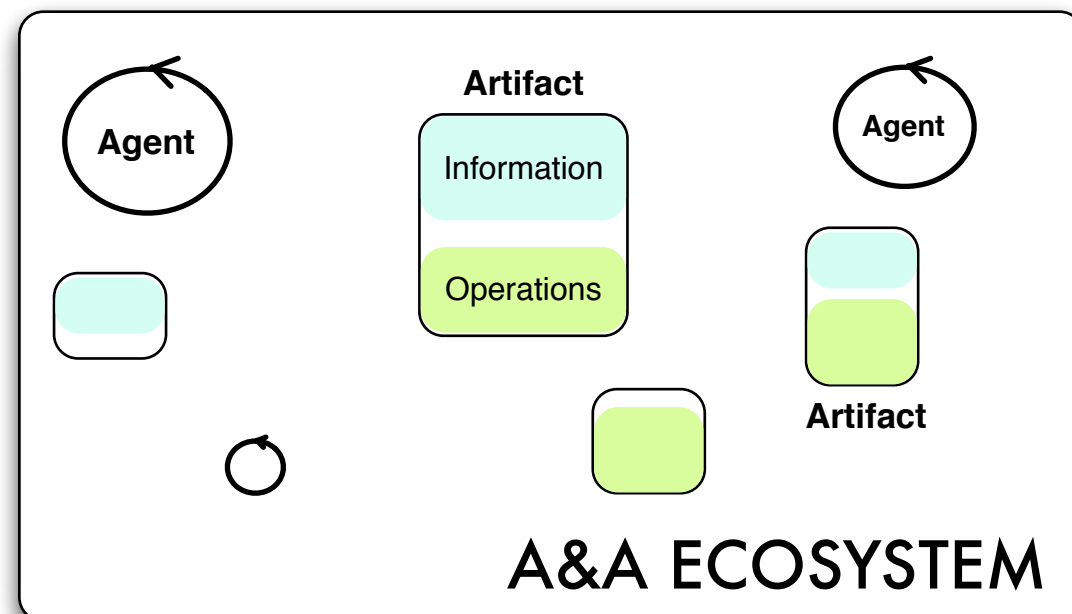
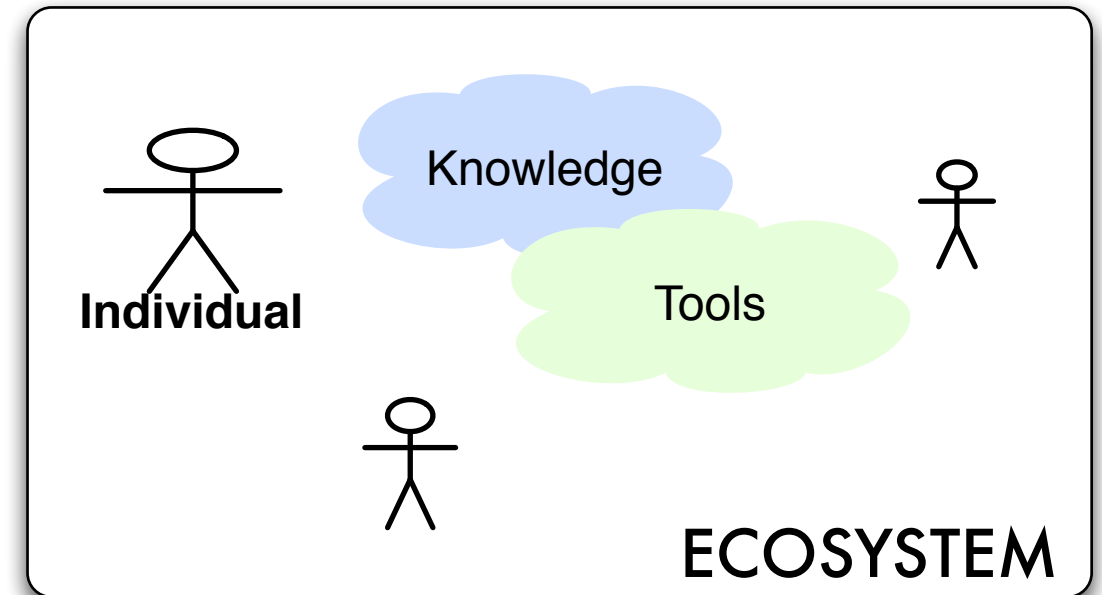
# A&A Ecosystems

An ecosystem is a natural unit consisting of all **plants, animals and micro-organisms (biotic factors)** in an area functioning together with all of the **non-living physical (abiotic) factors** of the environment.  
[Christopherson, RW (1996)]



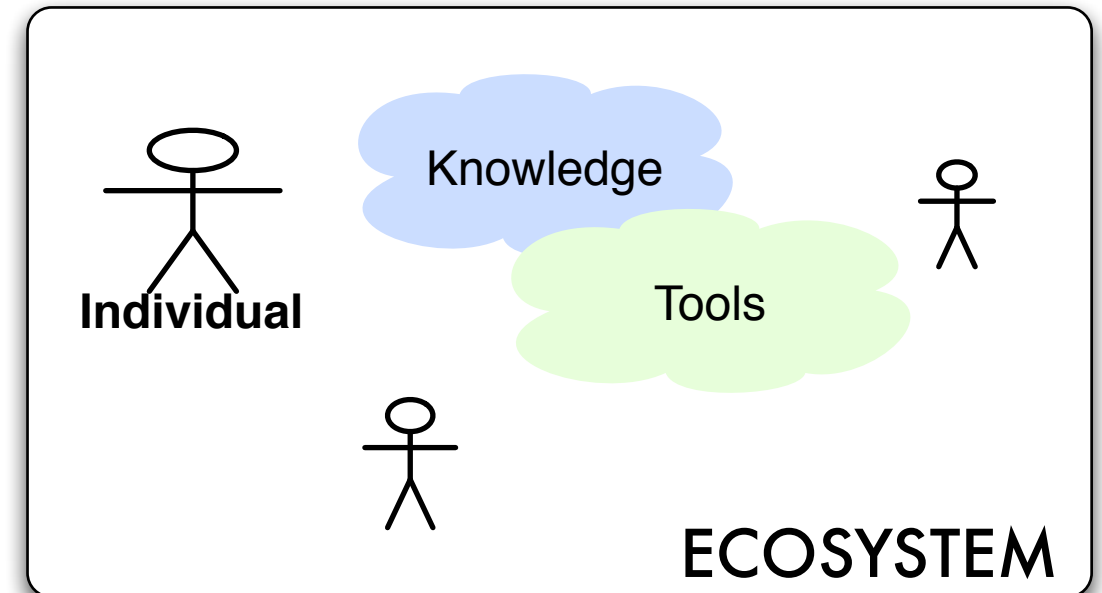
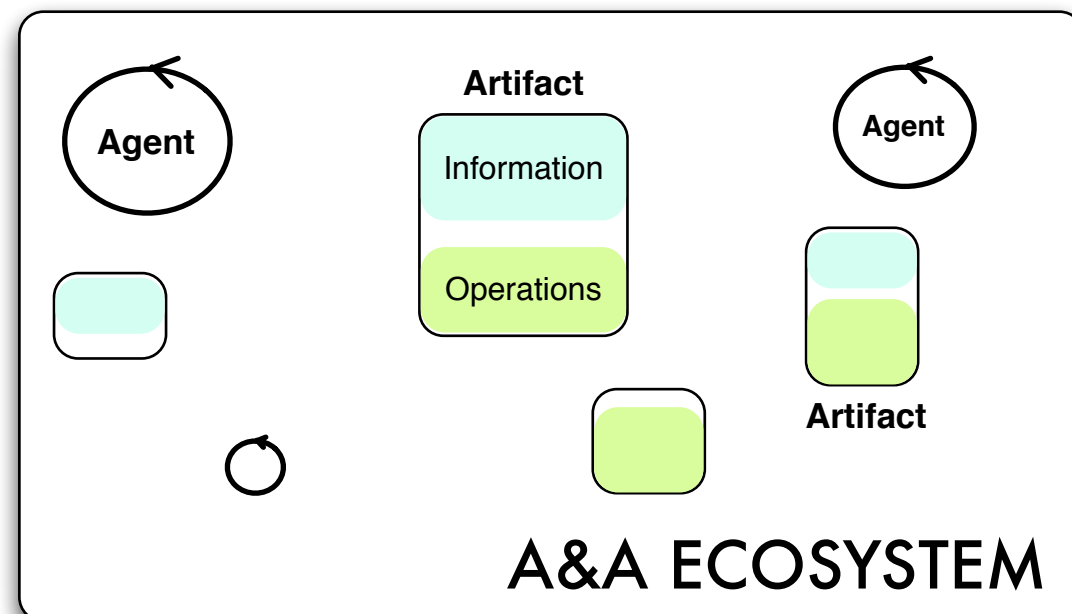
# A&A Ecosystems

An ecosystem is a natural unit consisting of all **plants, animals and micro-organisms (biotic factors)** in an area functioning together with all of the **non-living physical (abiotic) factors** of the environment.  
[Christopherson, RW (1996)]



# A&A Ecosystems

An ecosystem is a natural unit consisting of all **plants, animals and micro-organisms (biotic factors)** in an area functioning together with all of the **non-living physical (abiotic) factors** of the environment.  
[Christopherson, RW (1996)]



Agents as *intentional systems*

Artifact are supposed to be (not only) external component (but also) functional resources

## Functional Approach

- Representational Function
- Operational Function

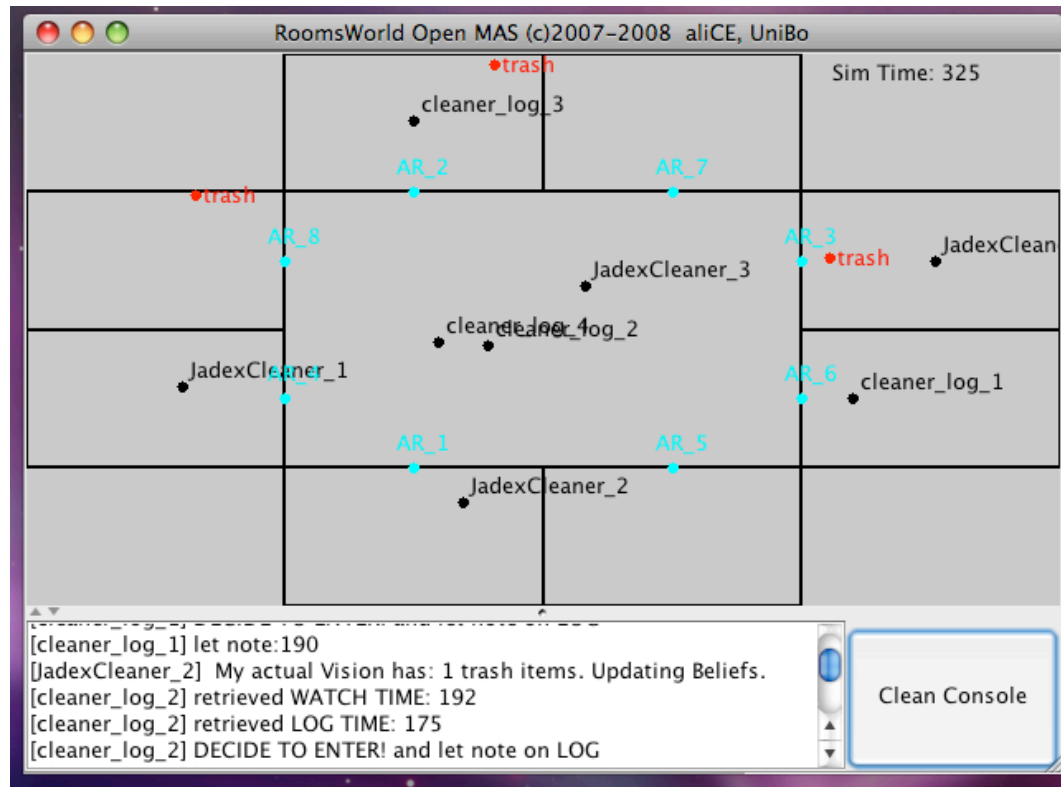
# Operational Function

- Operations are Artifacts *intended* purposes
  - “*Intended*” is in the mind of Artifact Designers
- From an **agent viewpoint**, Artifacts are non-autonomous but *automatic* devices providing functionalities exploitable as self contained operations:
  - Improving repertoire of actions, as additional means to achieve Goals
  - Externalise and distribute part of agent activities

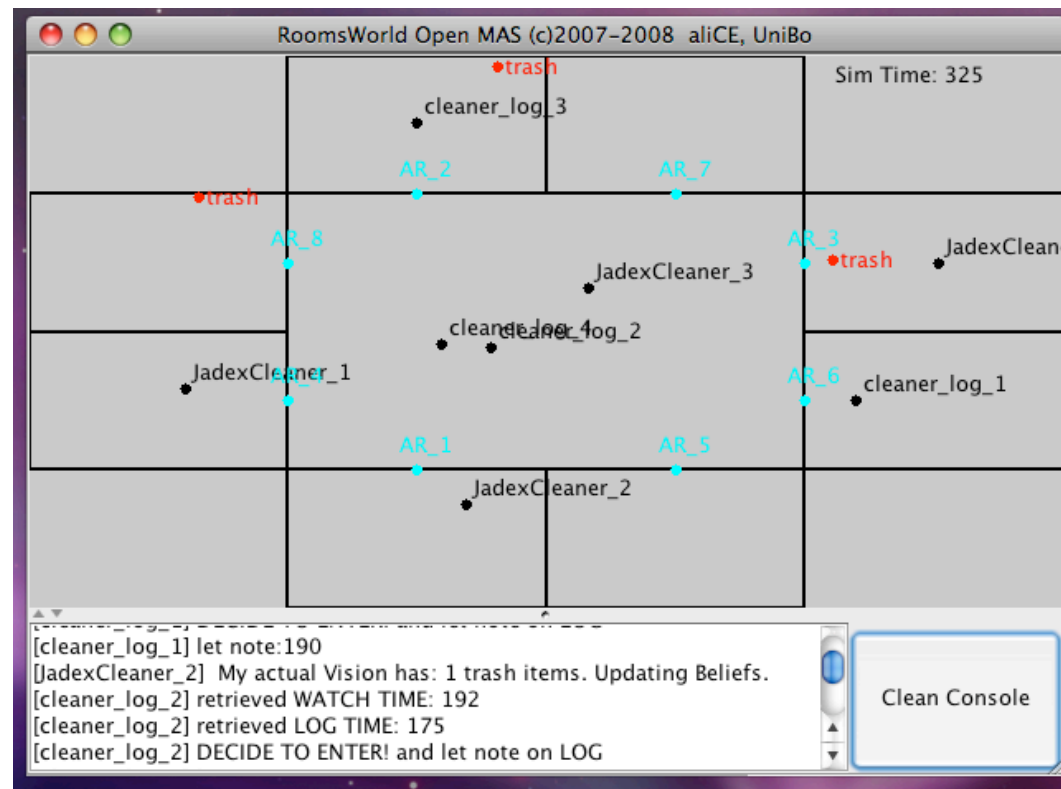
# Representational Function

- Artifacts include *machine-readable* Representations
  - Can maintain, make it observable, pre-process information
- From an **agent viewpoint**, Artifacts are *informational* units exploitable in a *situated* way:
  - As external repositories, automatically collecting and providing strategic knowledge
  - Additional memory, even shared between agent groups
  - Observable cues in order to highlight relevant information (*situated cognition*)

# RoomsWorld



# RoomsWorld

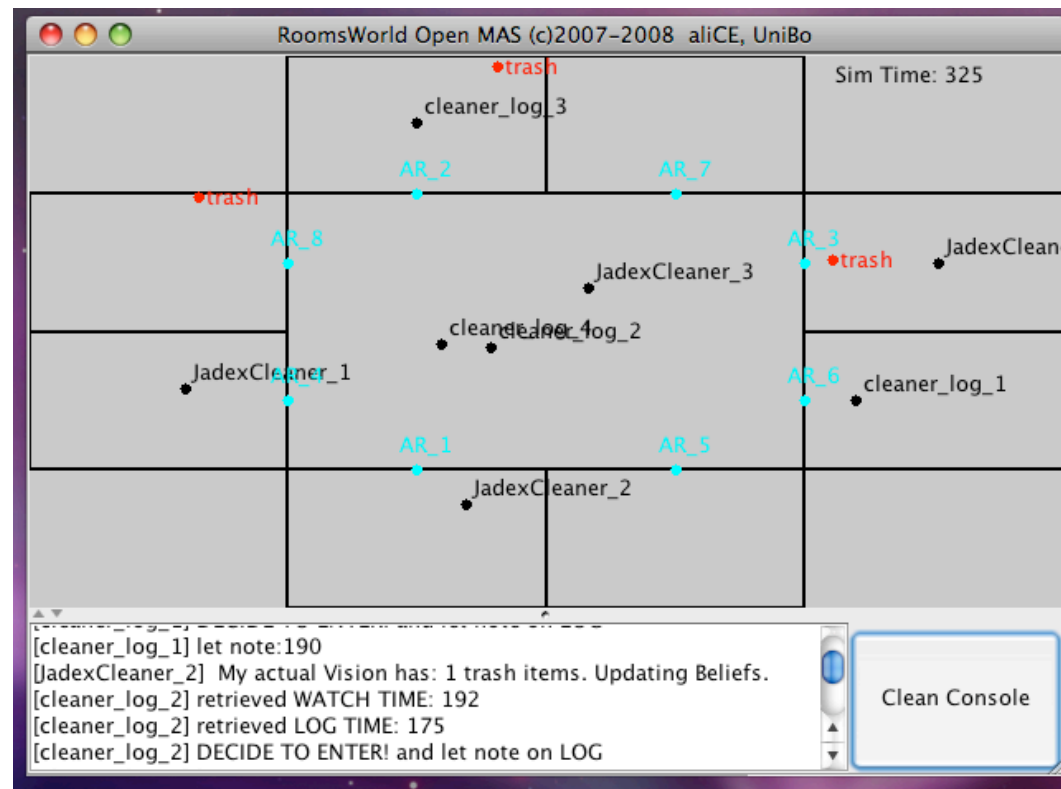


## GOAL: clean rooms

Agents have bounded range of sight  
i.e., can locate trash items only entering the room



# RoomsWorld



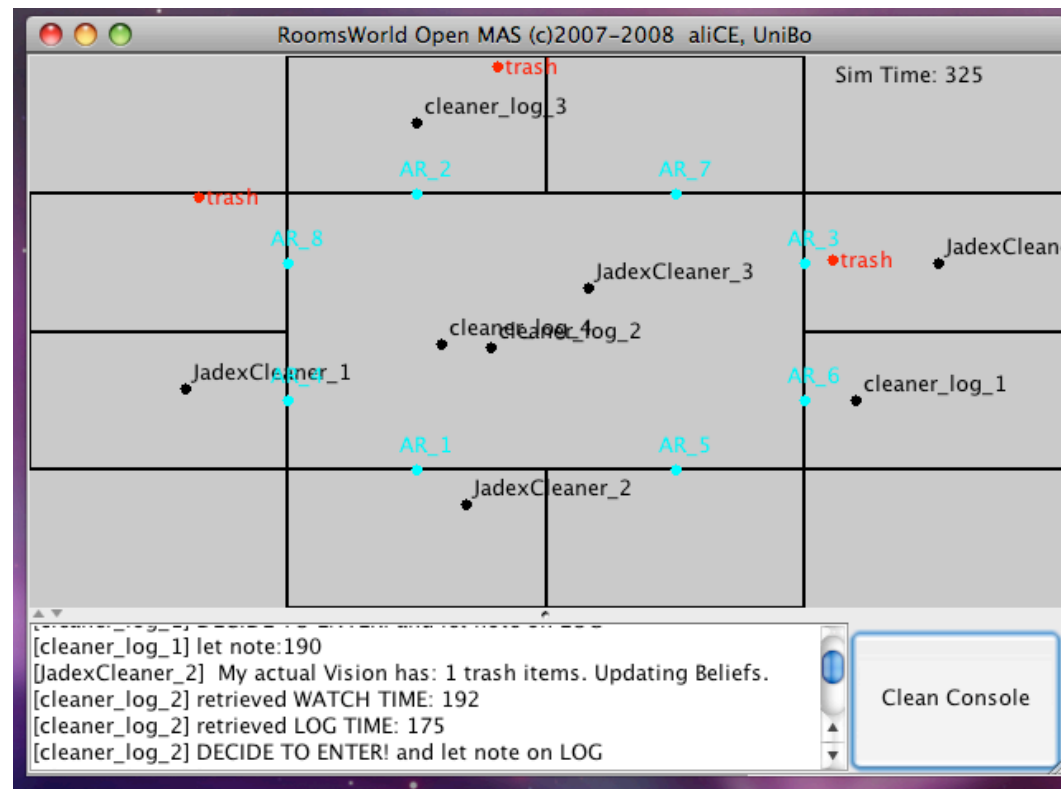
## GOAL: clean rooms

Agents have bounded range of sight  
i.e., can locate trash items only entering the room

To prevent wasting resources (*time*) agents  
need a good strategy to coordinate activities

Each agent should deliberate the room to visit  
*knowing and evaluating* the activities performed  
**by others**

# RoomsWorld



## GOAL: clean rooms

Agents have bounded range of sight  
i.e., can locate trash items only entering the room

To prevent wasting resources (*time*) agents  
need a good strategy to coordinate activities

Each agent should deliberate the room to visit  
*knowing and evaluating* the activities performed  
**by others**

The information about which room has already been  
cleaned is *relevant* for deciding the next course of actions:

## GOAL-SUPPORTING BELIEFS

# Introducing Cognitive Artifacts

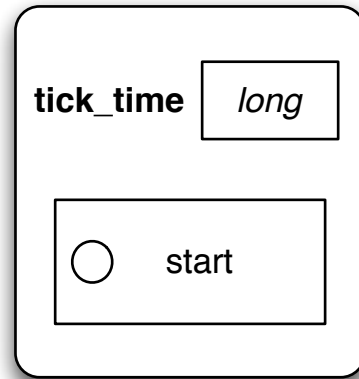
## WATCH

### OBSERVABLE PROPERTIES

**tick\_time** *long*

### USAGE INTERFACE

start()



```
public class Watch extends Artifact {

    @OBSPROPERTY long tick_time;
    private final long exp_length = 2500;

    @OPERATION void init(){
        tick_time=0;
    }

    @OPERATION void start() {
        try {
            nextStep("tick");
        } catch (Exception ex) {}
    }

    @OPSTEP(tguard=1000) void tick(){
        if (tick_time<exp_length){
            try {
                updateProperty("tick_time", ++tick_time);
                // signal to synchronize dwelling agents
                signal("grig_tick", tick_time);
                outSignal("link", new Op("setSimTime", tick_time ));
                nextStep("tick");
            } catch (Exception ex){}
        } else {
            signal("exp_end");
        }
    }
}
```

# Introducing Cognitive Artifacts

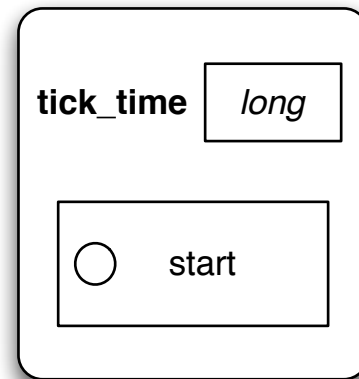
## WATCH

OBSERVABLE PROPERTIES

**tick\_time** *long*

USAGE INTERFACE

start()



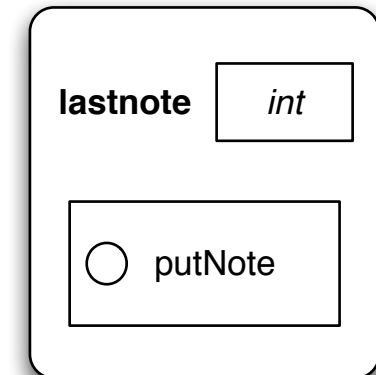
## LOG

OBSERVABLE PROPERTIES

**lastnote** *int*

USAGE INTERFACE

putNote(Object note)



```
public class Watch extends Artifact {

    @OBSPROPERTY long tick_time;
    private final long exp_length = 2500;

    @OPERATION void init(){
        tick_time=0;
    }

    @OPERATION void start() {
        try {
            nextStep("tick");
        } catch (Exception ex) {}
    }

    @OPSTEP(tguard=1000) void tick(){
        if (tick_time<exp_length){
            try {
                updateProperty("tick_time", ++tick_time);
                // signal to synchronize dwelling agents
                signal("grig_tick", tick_time);
                outSignal("link", new Op("setSimTime", tick_time ));
                nextStep("tick");
            } catch (Exception ex){}
        } else {
            signal("exp_end");
        }
    }
}
```

```
public class Log extends Artifact {

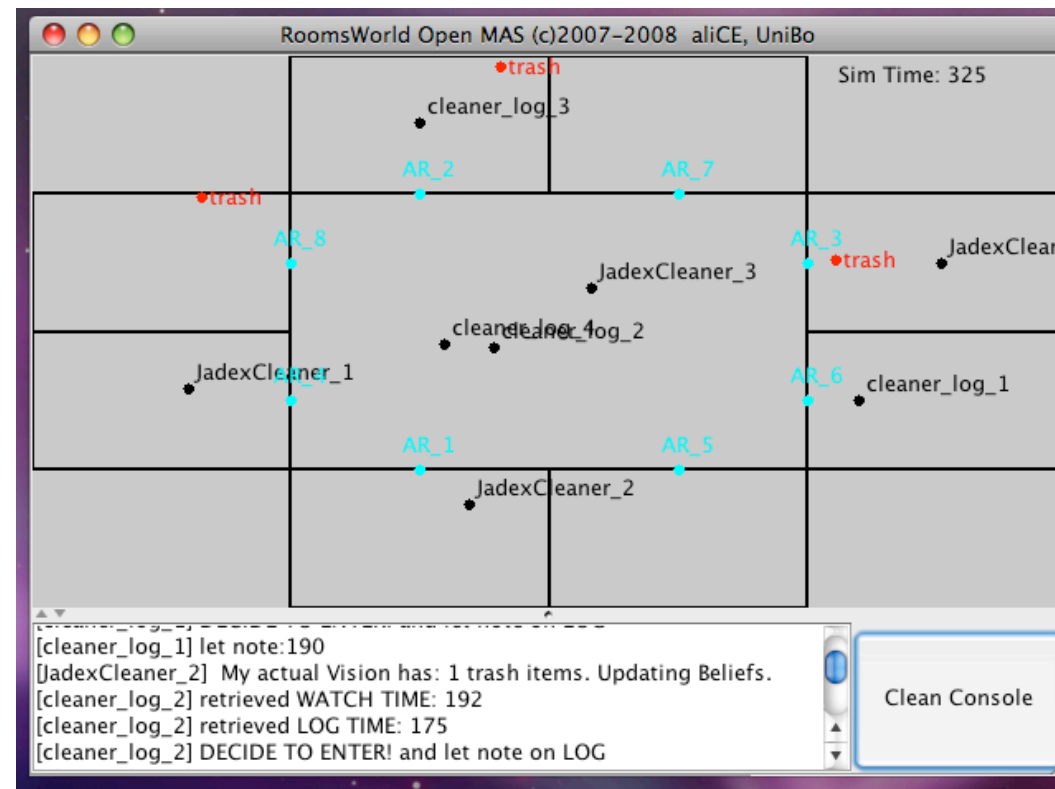
    private LinkedList<String> notes;
    private String name;
    private location loc;

    @OBSPROPERTY int lastnote;

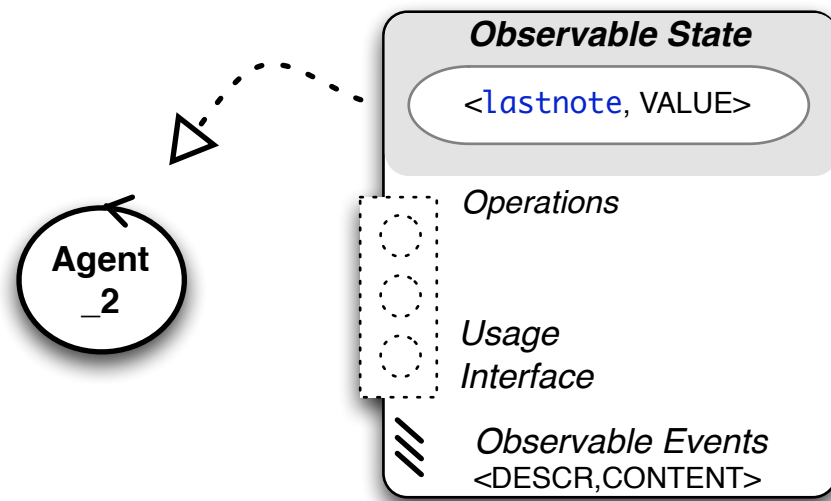
    @OPERATION void init(String n, location l){
        name=n; loc= l;
        notes = new LinkedList<String>();
    }

    @OPERATION void putNote( Object ts ){
        notes.addLast(ts.toString());
        lastnote = ((Integer)ts).intValue();
    }
}
```

# DEMO



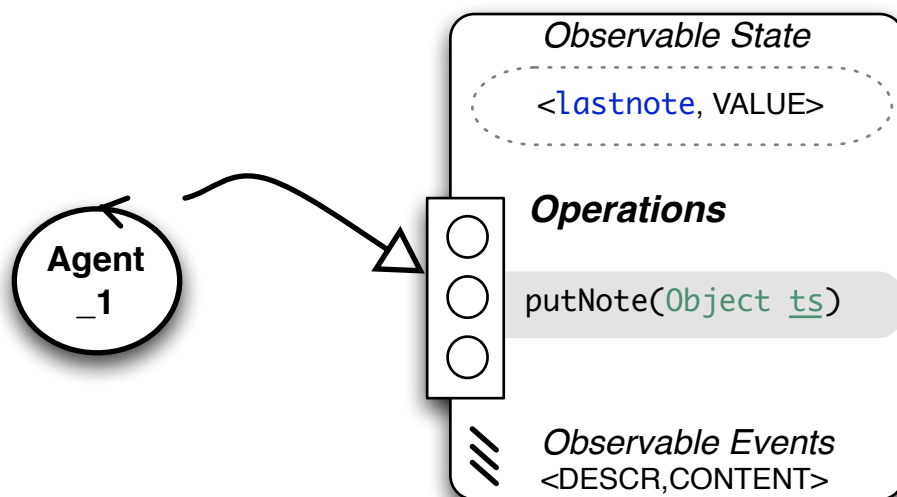
# Easing Deliberation (Jason)



[ ... retrieve IDLog and IDWatch ... ]

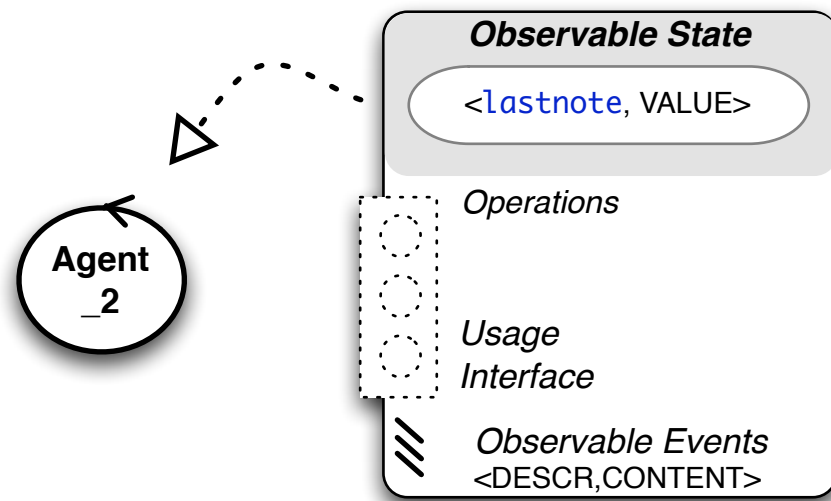
```
cartago.observeProperty(IDLog, lastnote(LogT) );
cartago.observeProperty(IDWatch, sim_time(Wt));
```

```
!decide(N, Wt, LogT).
```



```
...
cartago.lookupArtifact(LOG, LogID);
cartago.use(LogID, putNote(CurrentTime));
...
```

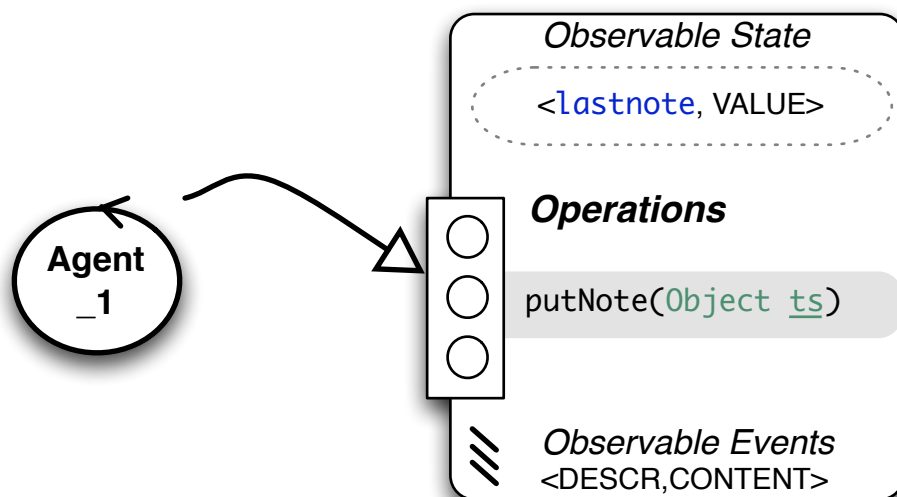
# Easing Deliberation (Jason)



[ ... retrieve IDLog and IDWatch ... ]

```
cartago.observeProperty(IDLog, lastnote(LogT));
cartago.observeProperty(IDWatch, sim_time(Wt));
```

```
!decide(N, Wt, LogT).
```



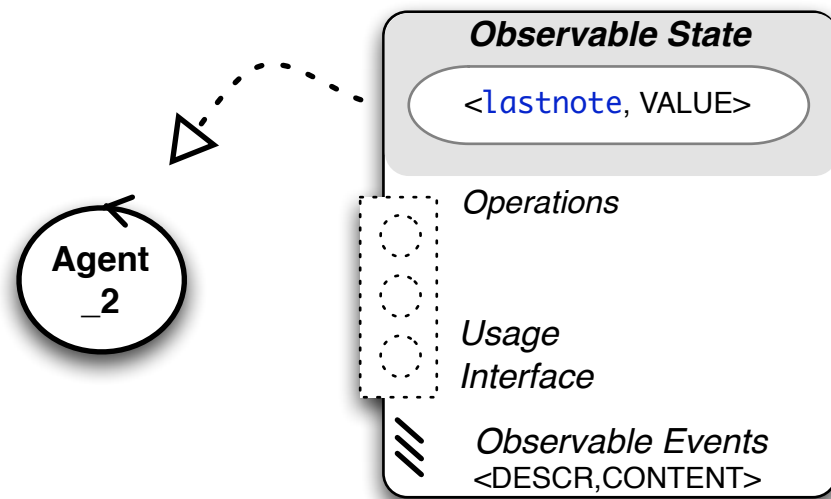
```
...
cartago.lookupArtifact(LOG, LogID);
cartago.use(LogID, putNote(CurrentTime));
...
```

```
+!observeLog(N)
  <- // [ ... retrieve IDLog and IDWatch ... ]
      cartago.observeProperty(IDLog, lastnote(LogT));
      cartago.observeProperty(IDWatch, sim_time(Wt));
      !decide(N, Wt, LogT).

+!decide(N, CurrentTime, LastTime)
  : (day(D) & (D < CurrentTime-LastTime)) | (LastTime=0)
  <- !log("DECIDE TO ENTER! Put a note on Log");
      // [ ... retrieve sensor S and Log name LogN ... ]
      cartago.lookupArtifact(LogN, IDLog);
      cartago.use(IDLog, putNote(CurrentTime), S);
      !enter(N).

+!decide(N, CurrentTime, LastTime)
  <- !log("RECONSIDERED INTENTIONS:EXPLORING!");
      -cleaned(N,_);
      +cleaned(N,LastTime);
      -targetRoom(_);
      !explore.
```

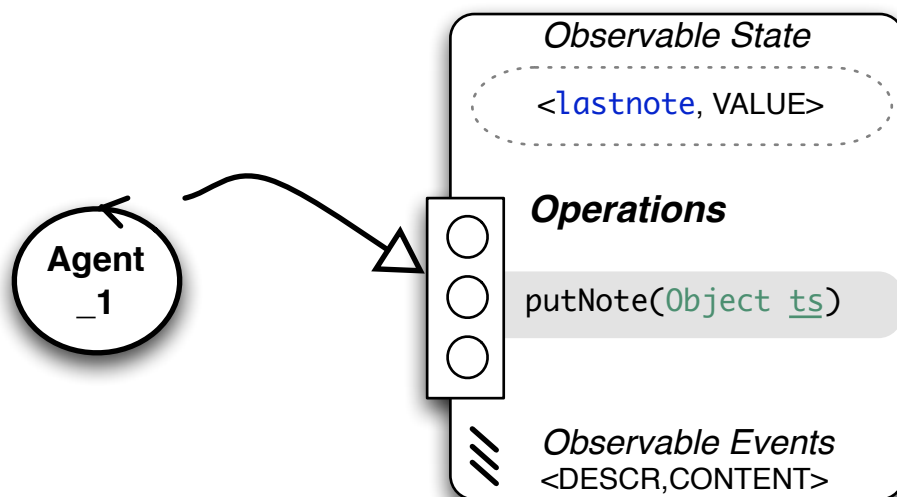
# Easing Deliberation (Jason)



[ ... retrieve IDLog and IDWatch ... ]

```
cartago.observeProperty(IDLog, lastnote(LogT));
cartago.observeProperty(IDWatch, sim_time(Wt));
```

```
!decide(N, Wt, LogT).
```



```
...
cartago.lookupArtifact(LOG, LogID);
cartago.use(LogID, putNote(CurrentTime));
...
```

```
+!observeLog(N)
  <- // [ ... retrieve IDLog and IDWatch ... ]
      cartago.observeProperty(IDLog, lastnote(LogT));
      cartago.observeProperty(IDWatch, sim_time(Wt));
      !decide(N, Wt, LogT).

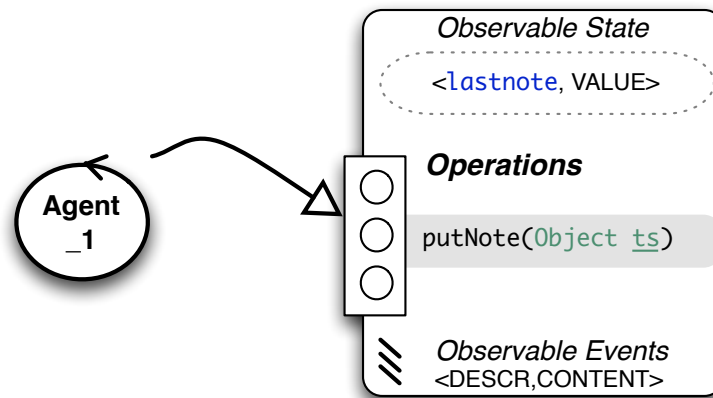
+!decide(N, CurrentTime, LastTime)
  : (day(D) & (D < CurrentTime-LastTime)) | (LastTime=0)
  <- !log("DECIDE TO ENTER! Put a note on Log");
      // [ ... retrieve sensor S and Log name LogN ... ]
      cartago.lookupArtifact(LogN, IDLog);
      cartago.use(IDLog, putNote(CurrentTime), S);
      !enter(N).

+!decide(N, CurrentTime, LastTime)
  <- !log("RECONSIDERED INTENTIONS:EXPLORING!");
      -cleaned(N,_);
      +cleaned(N,LastTime);
      -targetRoom(_);
      !explore.
```

Updating  
Goal-Supporting Beliefs

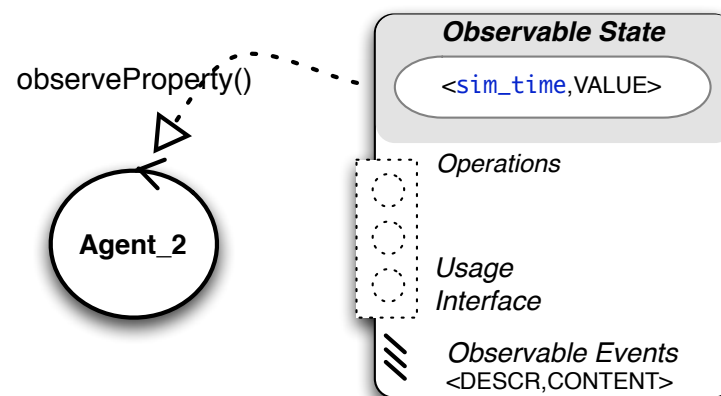


# Easing Deliberation (Jadex)



```

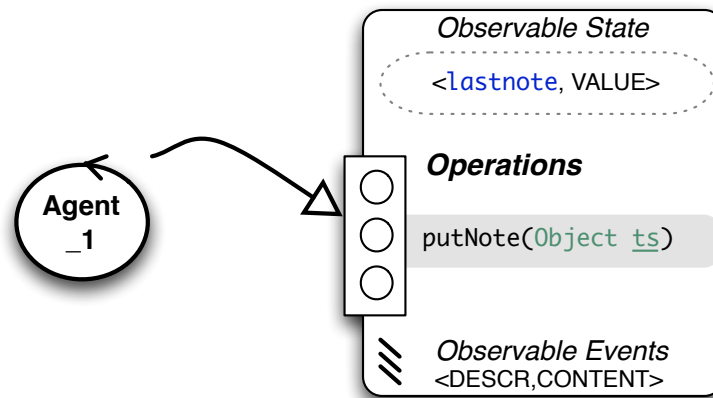
IGoal write = createGoal("use");
write.getParameter("artifact_name").setValue(target_log);
write.getParameter("sensor_name").setValue(sensorName);
Op op = new Op("putNote", time );
write.getParameter("op").setValue(op);
dispatchSubgoalAndWait(write);
    
```



```

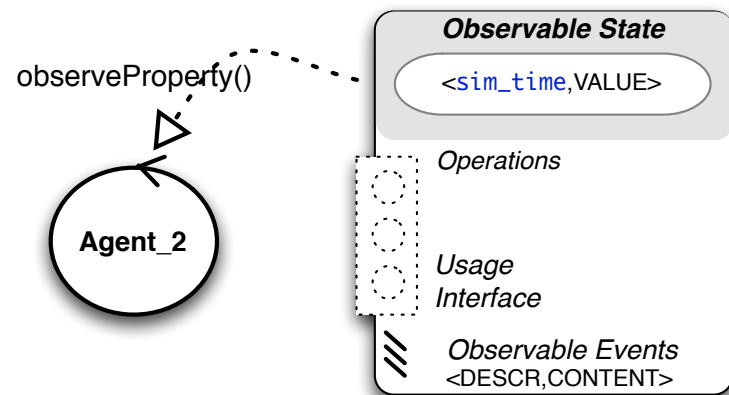
IGoal observeLog = createGoal("observe_property");
observeLog.getParameter("artifact_name").setValue(aName);
observeLog.getParameter("property").setValue("lastnote");
dispatchSubgoalAndWait(observeLog);
ArtifactProperty lastTime =
    (ArtifactProperty)observeLog.getParameter("value").getValue();
int lTime = (Integer)lastTime.getContent(0);
log(" observed lastTime:" + lTime + " on Log");
    
```

# Easing Deliberation (Jadex)



```

IGoal write = createGoal("use");
write.getParameter("artifact_name").setValue(target_log);
write.getParameter("sensor_name").setValue(sensorName);
Op op = new Op("putNote", time );
write.getParameter("op").setValue(op);
dispatchSubgoalAndWait(write);
    
```



```

IGoal observeLog = createGoal("observe_property");
observeLog.getParameter("artifact_name").setValue(aName);
observeLog.getParameter("property").setValue("lastnote");
dispatchSubgoalAndWait(observeLog);
ArtifactProperty lastTime =
    (ArtifactProperty)observeLog.getParameter("value").getValue();
int lTime = (Integer)lastTime.getContent(0);
log(" observed lastTime:" + lTime + " on Log");
    
```

```

<performgoal name="enter_room">
  <creationcondition>
    ($beliefbase.tick_time - $beliefbase.last_time) >
    $beliefbase.day
  </creationcondition>
  <deliberation cardinality="1">
    <inhibits ref="reconsider_room"/>
  </deliberation>
</performgoal>
    
```

```

<performgoal name="reconsider_room">
  <creationcondition>
    ($beliefbase.trashes.length == 0 &&
    $beliefbase.my_room == $beliefbase.target_room)
    ||
    (($beliefbase.tick_time > 0) &&
    ($beliefbase.tick_time - $beliefbase.last_time)
    <= $beliefbase.day)
  </creationcondition>
</performgoal>
    
```

# Cognitive Artifacts

The notion is related to agents able to bring about representational and operational functions

# Cognitive Artifacts

The notion is related to agents able to bring about representational and operational functions

- To provide Relevant, Strategic Information to Single-Agent
  - Easy reasoning about goals: *goal supporting beliefs*
  - Simplifying agent choices and easing deliberation processes

# Cognitive Artifacts

The notion is related to agents able to bring about representational and operational functions

- To provide Relevant, Strategic Information to Single-Agent
  - Easy reasoning about goals: *goal supporting beliefs*
  - Simplifying agent choices and easing deliberation processes
- To distribute Information in Multi-Agent
  1. Across *Agents*: organise and make available relevant information as permanent side-effect of artifact use (modification of artifact state)
  2. Across *Platforms*: mediated interactions
  3. Across *Time*: hold strategic information persistent over agent presence
  4. Across *Space*: no need for agents mutual presence within a location or heavy message exchange protocols

# Concluding Remarks

Artifacts can be *cognitively* used once their representational and operational contents are mapped into reasoning processes

- A&A interaction is approached at a programming level
- The problem is now how to make agents aware about artifacts which are *not known* at desing time

Artifact Representational Contents (i.e. manuals, properties, interfaces) has to be *gounded* with agents *epistemic* (beliefs) and *motivational* (goals) states.

# Two more Steps towards *Extended Mind*

- **Epistemic Reasoning** need to face with Artifact discovery
  - *Read, learn, and match* their Representational contents
  - Information + Control for Supervising externalized activities
- Operational contents has to be included in **Practical Reasoning** (to achieve goals)
  - By changing the actions required for achieving a goal, artifact operations change means-end reasoning (planning)

# end

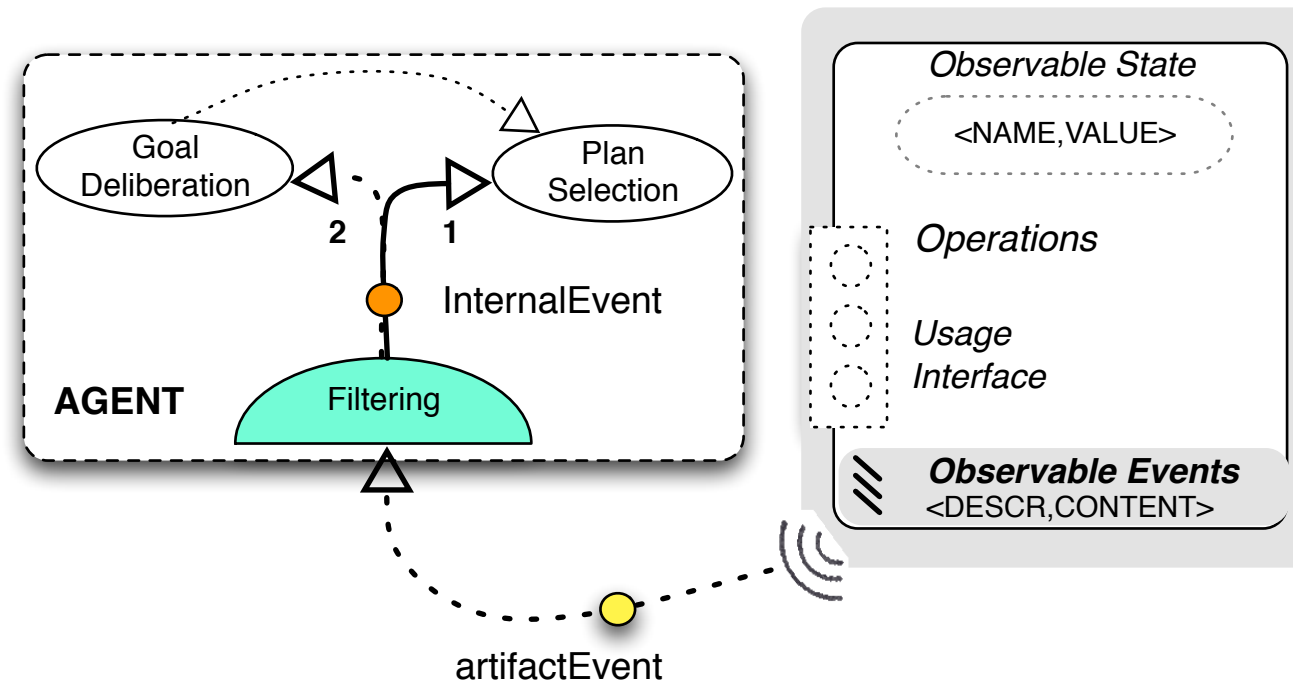


# Controlling Externalized Activities

A pivotal aspect is for activities required for the control and the coordination of externalized processes

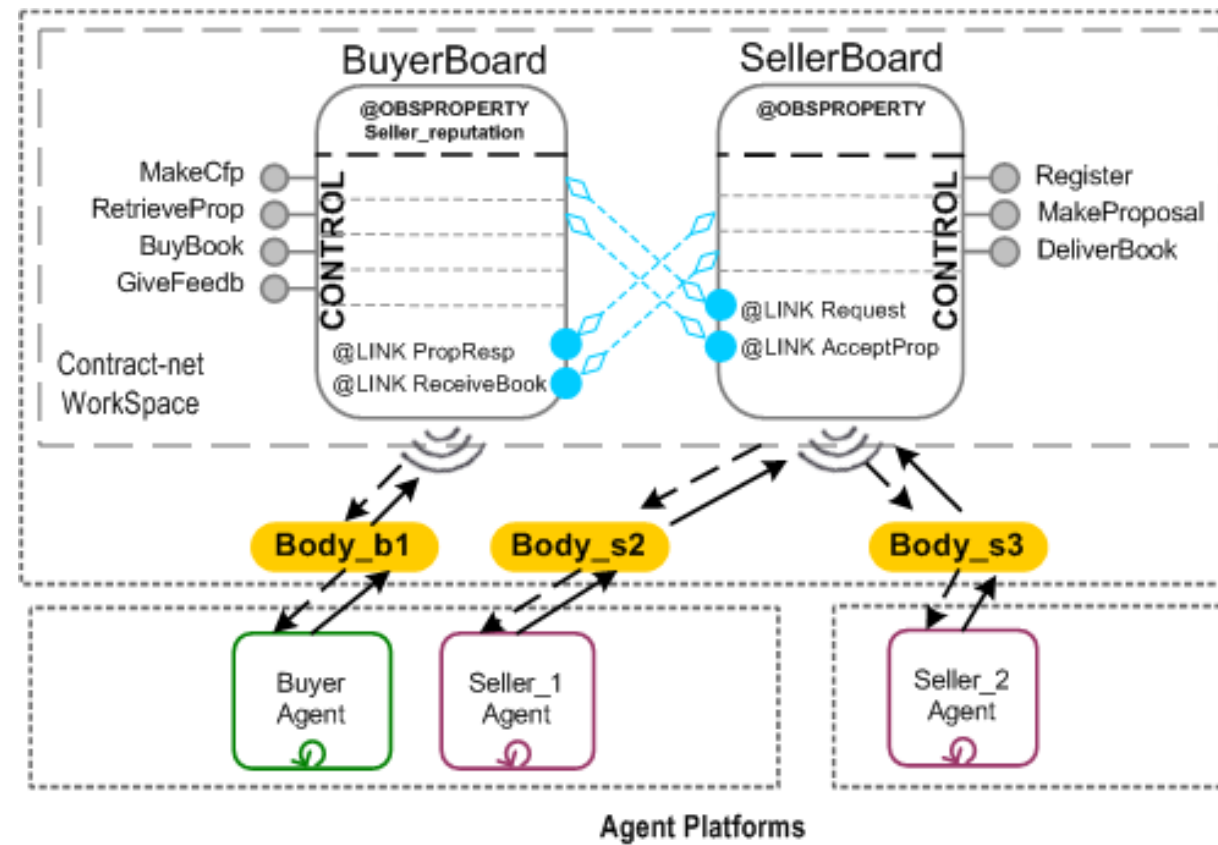
- Relations from cognitive agents to cognitive artifacts are bidirectional.
- External events coming from an artifact can be integrated at an architectural level by automatically promoting such events as signals requiring servicing
  - to be addressed to the reasoning processes.

# Agents as Dual Systems



1. Once encoded, the controlling a given interaction can *bypass* practical reasoning
  - Routinized activities and *reactive* behavior
  - No need for expansive deliberation processes
2. Particular events can be used for signalling something wrong
  - Mismatch on filtering rules
  - Reconsider Intention through reasoning

# Contract-Net



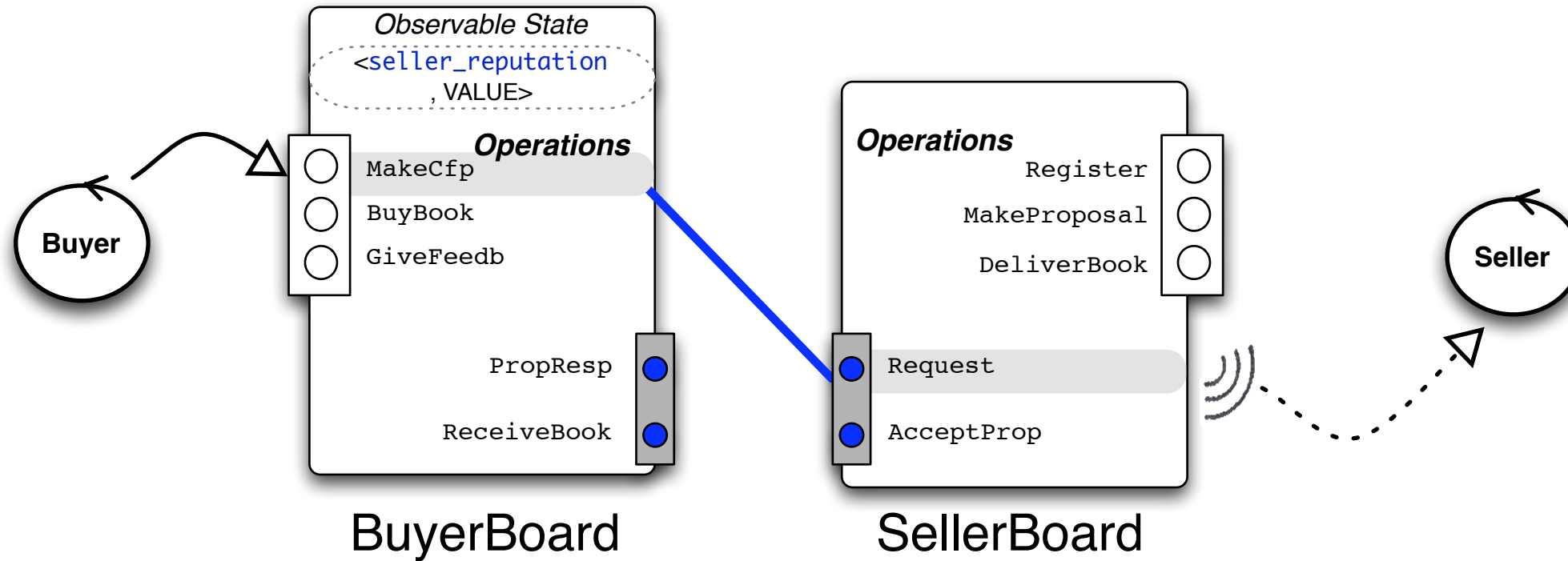
- Buyer and Seller mediate their interactions in a Contract-Net Workspace
- 2 Artifacts, *BuyerBoard* and *SellerBoard*, are linked to propagate each other relevant information

# Goal Directed Interaction

- Buyer adopt the Goal to buy a given book and initiate a call-for-proposal upon the BuyerBoard
- Goal directed agents can reason in terms of declarative Goals which have a proper representation and lifecycle
  - Declarative Goals
  - Deliberation is Independent from intention selection
  - Twofold reasoning processes for *adopting Goals* and *selecting Plans*

```
<achievegoal name="purchase_book" recur="true" recurdelay="10000">  
  <parameter name="order" class="Order">  
    <bindingoptions>$beliefbase.initial_orders</bindingoptions>  
  </parameter>  
  <unique/>  
  <creationcondition>  
    $beliefbase.initial_orders!=null &&  
    $beliefbase.sellers.length > 0  
  </creationcondition>  
  <targetcondition>  
    Order.DONE.equals($goal.order.getState())  
  </targetcondition>  
</achievegoal>
```

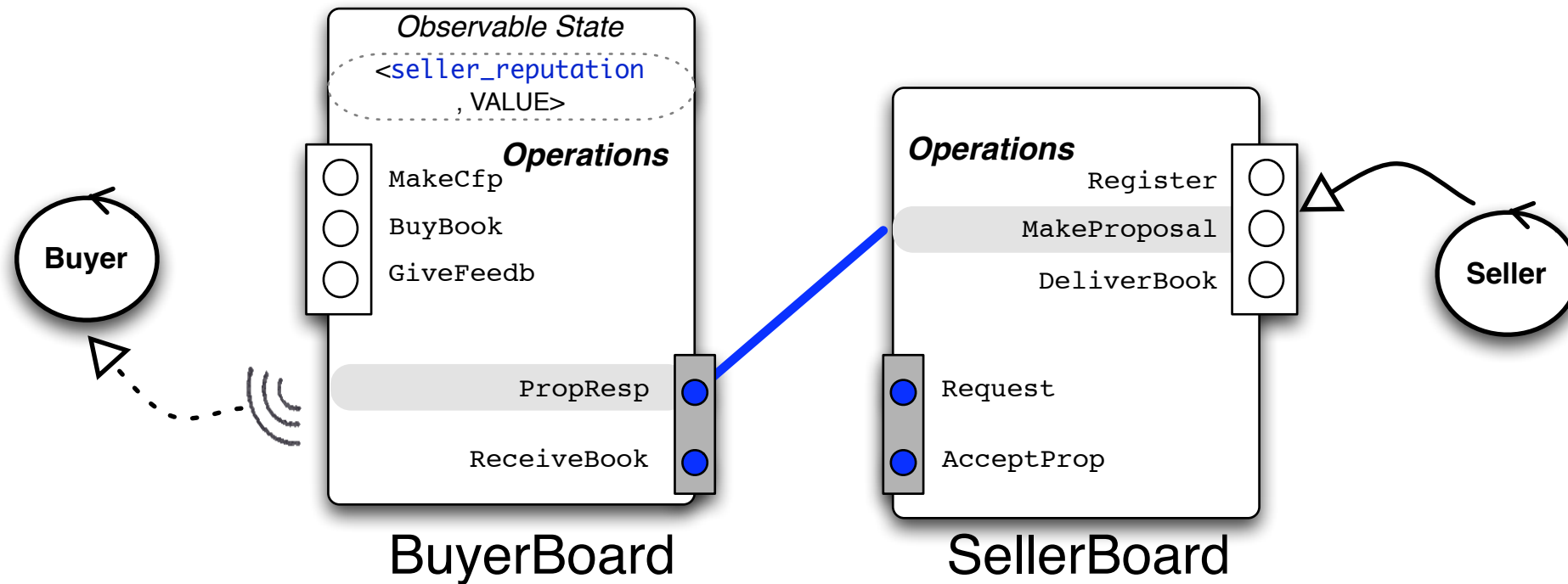
# Contract Net



```
//2. USE BuyerBoard to INITIATE CFP
IGoal initiate = createGoal("use");
initiate.getParameter("artifact_name").setValue(aName);
initiate.getParameter("sensor_name").setValue(sensorName);
op = new Op("MakeCfp", agentName, cfp, cfp_info,
    new AgentIdentifier(sellers));
initiate.getParameter("op").setValue(op);
dispatchSubgoalAndWait(initiate);
```

```
<!-- Participate Negotiation -->
<plan name="initiating_event">
  <parameter name="aName" class="String">
    <value>"SellerBoard"</value>
  </parameter>
  <body>new MakeProposalAction()</body>
  <trigger>
    <internalevent ref="artifact_event">
      <match>
        ((String)$event.label).equals("proposal_for_seller")
      </match>
    </internalevent>
  </trigger>
</plan>
```

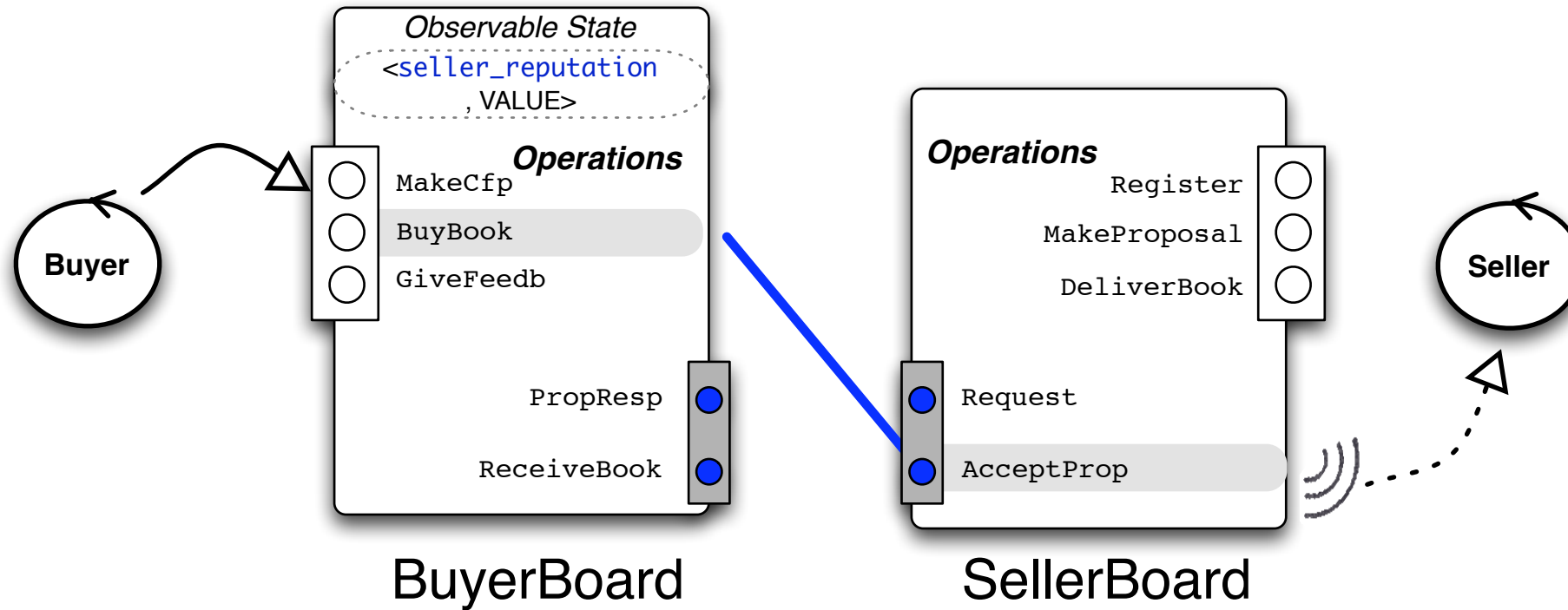
# Contract Net



```
<!-- Bid received -->
<plan name="receive_bid">
  <parameter name="aName" class="String">
    <value>"BuyerBoard"</value>
  </parameter>
  <body>new EvaluateBidAction()</body>
  <trigger>
    <internalevent ref="artifact_event"/>
    <match>
      ((String)$event.label).equals("response_from_seller")
    </match>
    </internalevent>
  </trigger>
</plan>
```

```
//2. MAKE A BID on SellerBoard ARTIFACT
IGoal make_proposal = createGoal("use");
make_proposal.getParameter("artifact_name").setValue(aName);
make_proposal.getParameter("sensor_name").setValue(sensorName);
Op op = new Op("MakeProposal", agentName, "buyer", cfp, prop, info);
make_proposal.getParameter("op").setValue(op);
dispatchSubgoalAndWait(make_proposal);
```

# Contract Net

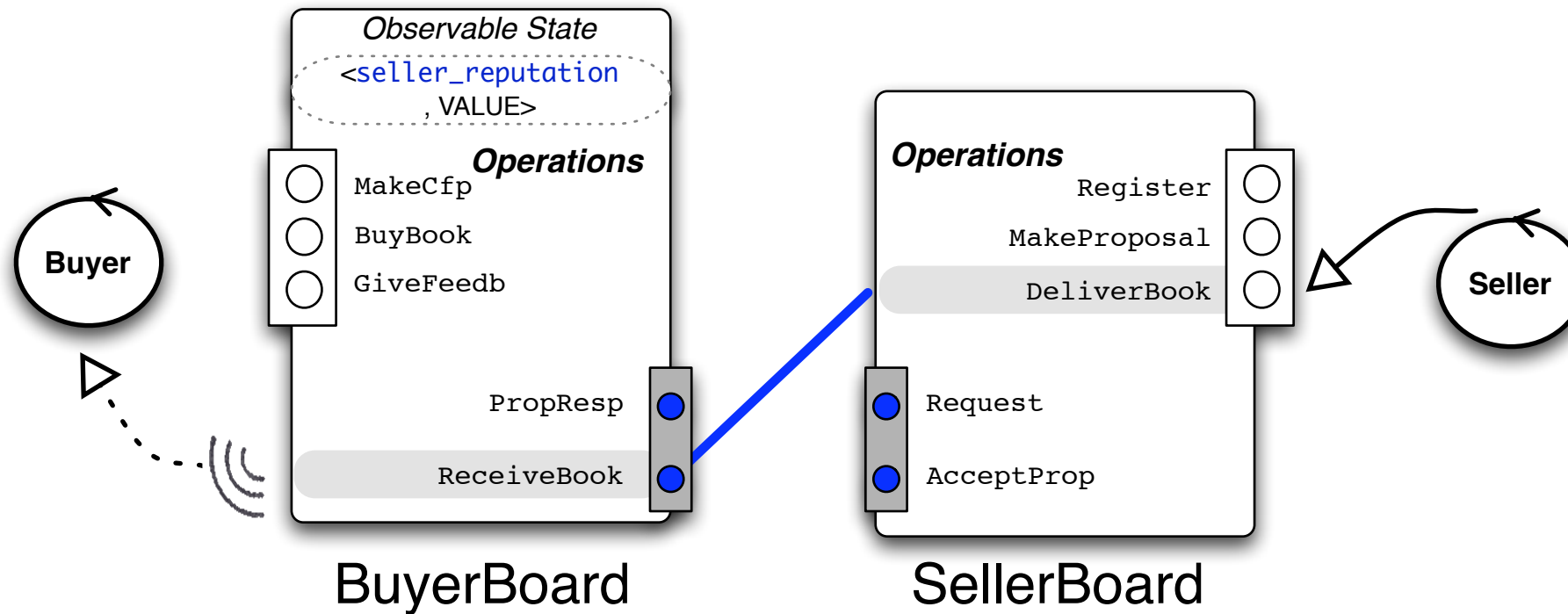


```
//4. USE PURCHASE ACTION on BuyerBoard ARTIFACT using
//@OPERATION void BuyBook(Object AgentName, Object cfp, Object SellName)
IGoal purchase = createGoal("use");
purchase.getParameter("artifact_name").setValue("BuyerBoard");
Op op = new Op("BuyBook", getAgentName(),
    p.getCfpInfo(), p.getAgSeller());
purchase.getParameter("op").setValue(op);
dispatchSubgoalAndWait(purchase);
```

```
<plan name="deliver_book">
  <parameter name="aName" class="String">
    <value>"participant"</value>
  </parameter>
  <body>new ExecuteOrderAction()</body>
  <trigger>
    <internalevent ref="artifact_event">
      <match>
        ((String)$event.label).equals("accepted_seller")
      </match>
    </internalevent>
  </trigger>
</plan>
```



# Contract Net



```
<!-- Book delivered -->
<plan name="receive_book">
  <parameter name="aName" class="String">
    <value>"BuyerBoard"</value>
  </parameter>
  <body>new FinalizeCnpAction()</body>
  <trigger>
    <internalevent ref="artifact_event">
      <match>
        ((String)$event.label).equals("cnp_end_buyer")
      </match>
    </internalevent>
  </trigger>
</plan>
```

```
// FINALIZE ORDER BY USING
// @OP DeliverBook on SellerBoard ARTIFACT
IGoal execute_order = createGoal("use");
execute_order.getParameter("artifact_name").setValue("SellerBoard");
execute_order.getParameter("sensor_name").setValue(sensorName);
Op op = new Op("DeliverBook", agentName, title, outObj);
execute_order.getParameter("op").setValue(op);
dispatchSubgoalAndWait(execute_order);
```





# Timer

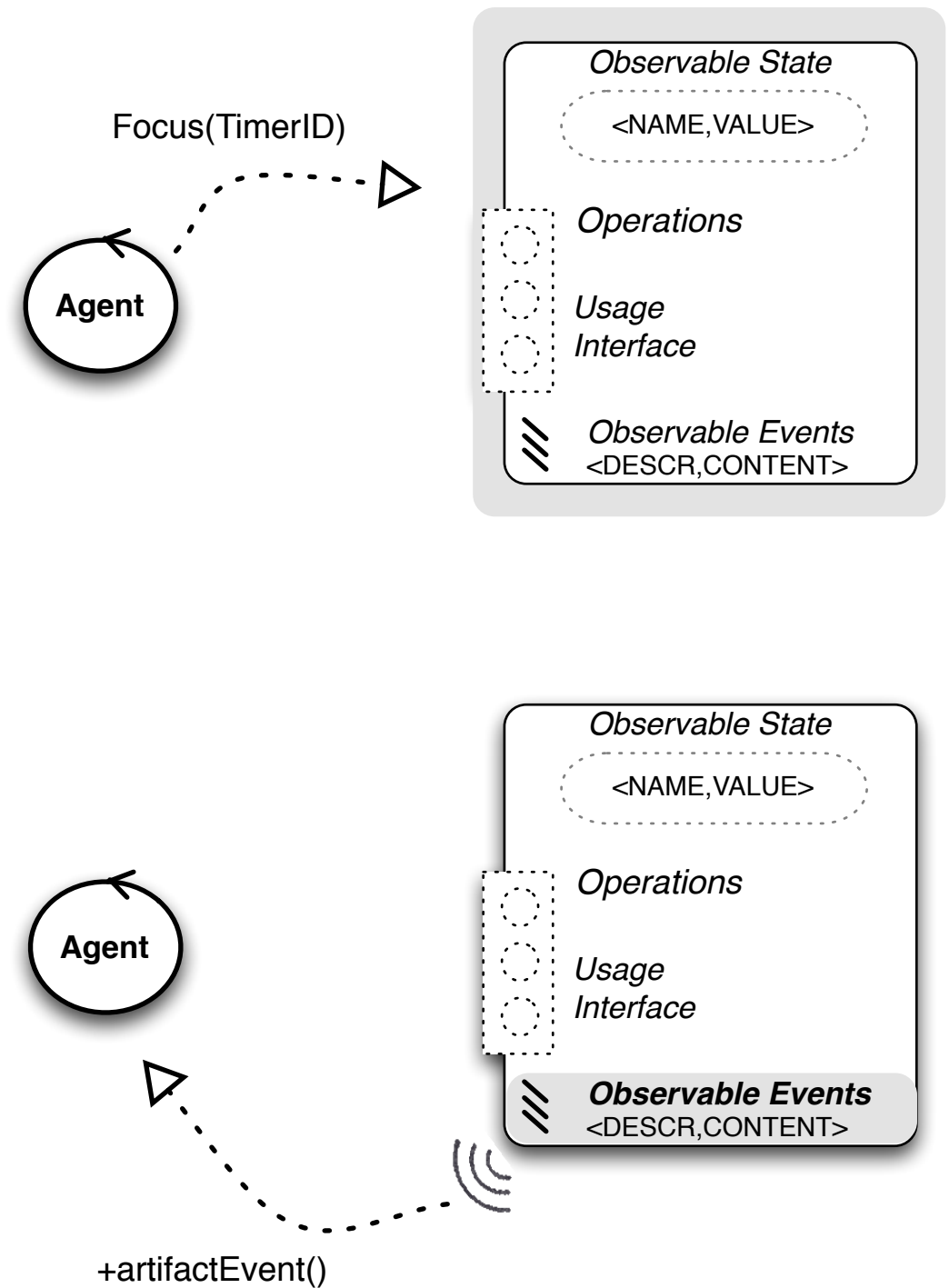
```
public class Timer extends Artifact {

    @OBSPROPERTY long tick_time;
    private final long exp_length = 2500;

    @OPERATION void init(){
        tick_time=0;
    }

    @OPERATION void start() {
        try {
            nextStep("tick");
        } catch (Exception ex) {}
    }

    @OPSTEP(tguard=1000) void tick(){
        if (tick_time<exp_length){
            try {
                updateProperty("tick_time", ++tick_time);
                // signal to synchronize dwelling agents
                signal("grig_tick", tick_time);
                outSignal("link", new Op("setSimTime", tick_time ));
                log(" TICK TIME: "+ tick_time);
                nextStep("tick");
            } catch (Exception ex){}
        } else {
            log("experiment_end!!");
            signal("exp_end");
        }
    }
}
```



# A Watch

```
public class Watch extends Artifact{

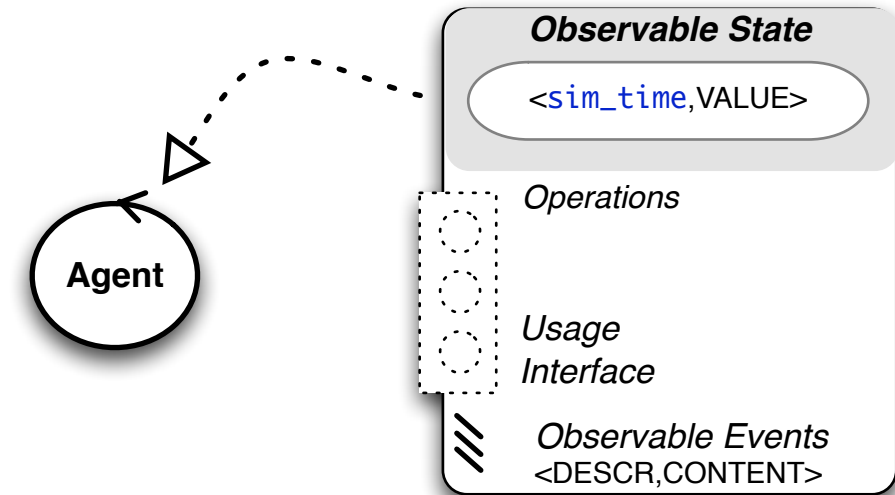
    @OBSPROPERTY long sim_time;
    private final long exp_length = 303;

    @OPERATION void init(){
        sim_time=0;
    }

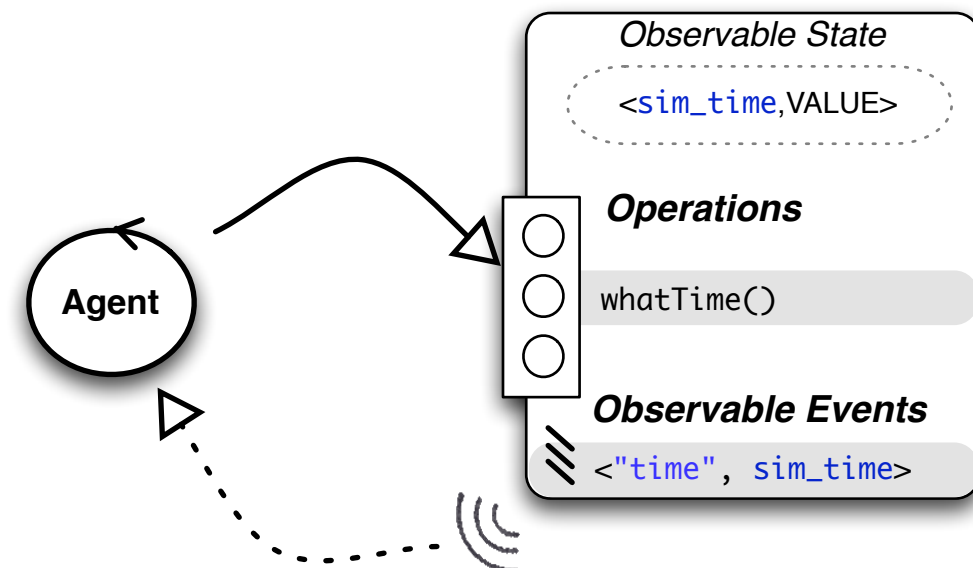
    @OPERATION void start() {
        try {
            nextStep("tick");
        } catch (Exception ex) {}
    }

    @OPSTEP(tguard=1000) void tick(){
        if (sim_time<exp_length){
            try {
                updateProperty("sim_time", ++sim_time);
                // signal to synchronize dwelling agents
                signal("tick", sim_time);
                outSignal("link", new Op("setSimTime",tick_time));
                nextStep("tick");
            } catch (Exception ex){}
        } else {
            log("experiment_end!!");
            signal("exp_end");
        }
    }

    @OPERATION void whatTime() {
        //log("time is: "+ sim_time);
        signal("time", sim_time);
    }
}
```



```
?artifactBel(watch, IDWatch);
....
cartago.observeProperty(IDWatch, sim_time(Wt));
...
```



```
?mySensor(S);
..
cartago.use(IDWatch, whatTime, S);
cartago.sense(S, time(Wt) );
...
```