ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
SEDE DI CESENA

aliCE

WOA, Palermo Novembre 2008

From Agents to Artifacts Back and Forth *Operational and Doxastic Use of Artifacts in MAS*

Michele Piunti(1,2), Alessandro Ricci (1)

(1) aliCE group at DEIS, Università di Bologna, Cesena

(2) Istituto Scienze e Tecnologie della Cognizione, ISTC-CNR, Roma

[michele.piunti, alessandro.ricci]@unibo.it

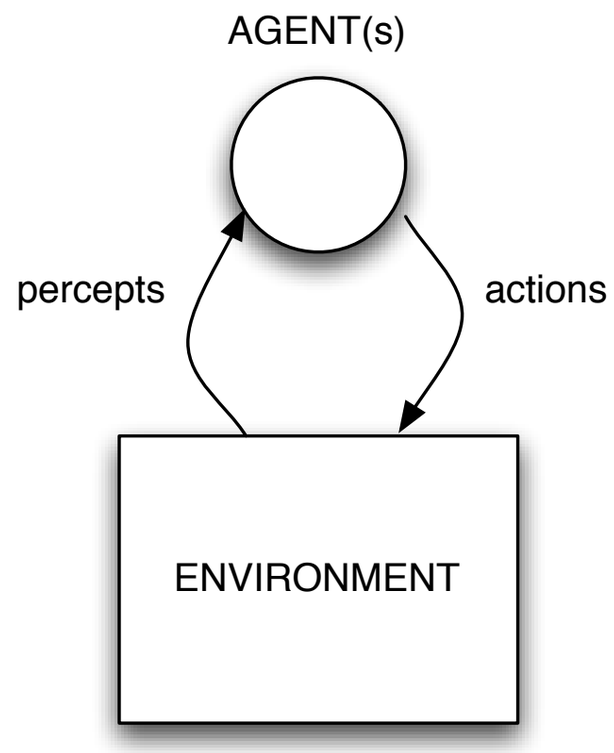
presented by

Andrea Omicini

OUTLINE

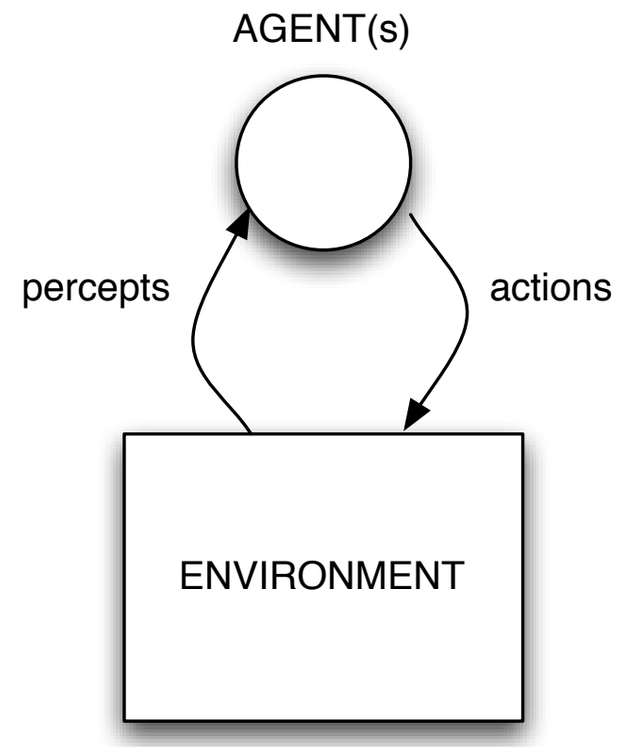
- Rationale of Agents and Artifacts (A&A) approach
- CArtAgO and Artifact programming model
- Agents and Artifacts in the Loop of Cognitive Interactions
- Example: Producers-Consumers
- Conclusions

THE ROLE OF ENVIRONMENT IN MAS



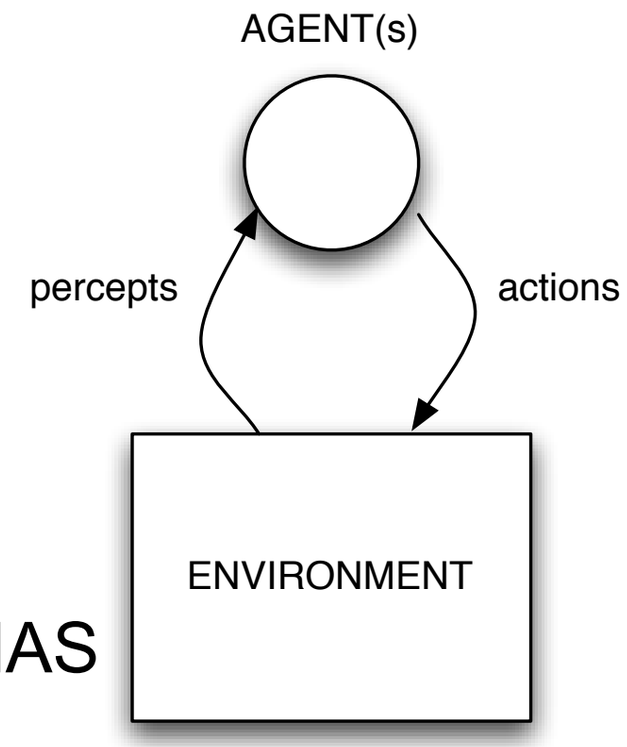
THE ROLE OF ENVIRONMENT IN MAS

- "Traditional" (D)AI / agent / MAS view
  - the target of agent actions and source of agents perception
  - something out of MAS design / engineering



THE ROLE OF ENVIRONMENT IN MAS

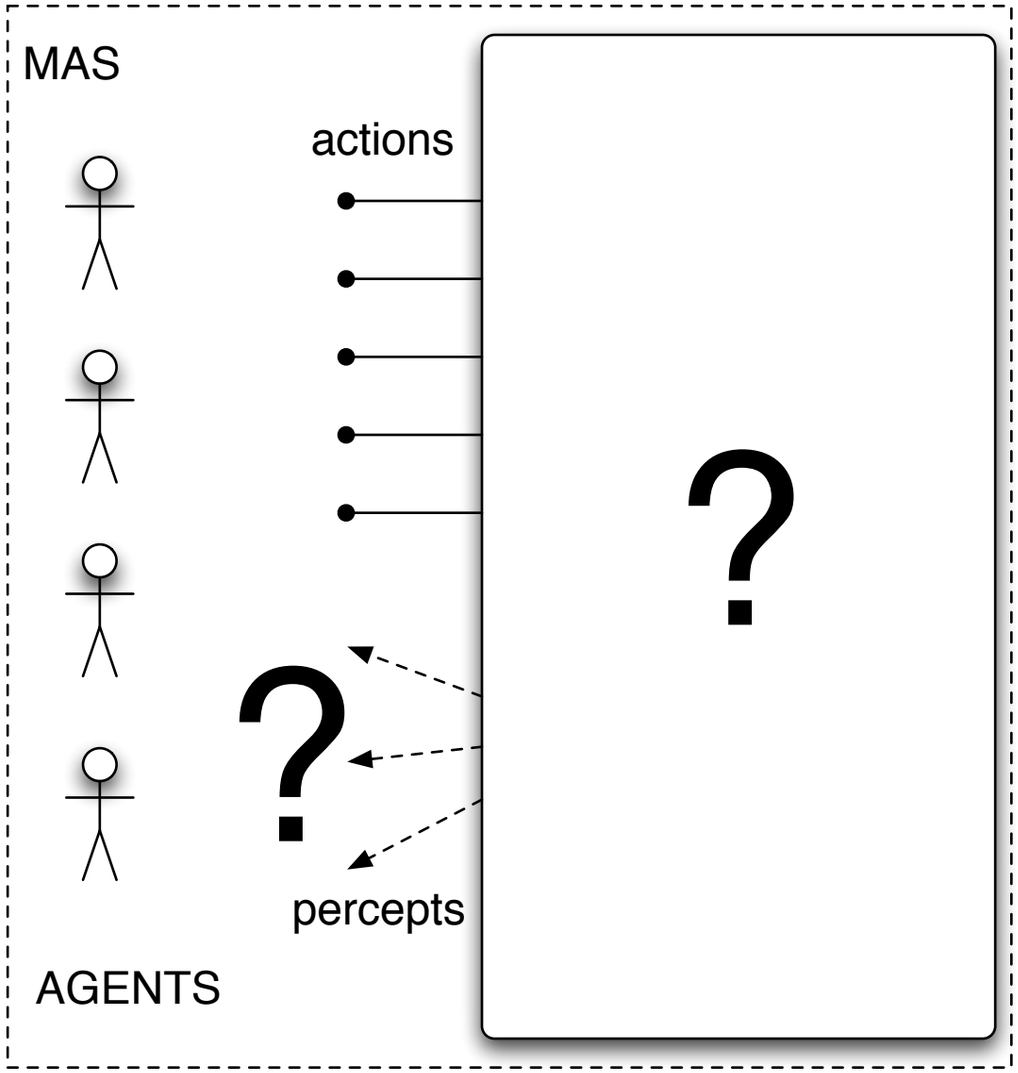
- "Traditional" (D)AI / agent / MAS view
  - the target of agent actions and source of agents perception
  - something out of MAS design / engineering



- New perspective in recent works
  - environment as a first-class issue in engineering MAS
    - mediating interaction among agents
    - encapsulating functionalities and services for enabling/ managing such interactions
      - coordination, organisation, security,...

ENVIRONMENT IN MAS PROGRAMMING ?

PERCEPTION IN AUTONOMOUS AGENTS ?



ENVIRONMENT MODELS IN MAS

ENVIRONMENT MODELS IN MAS

- Environment as monolithic / centralised block
  - defining agent (external) actions
    - typically a static list of actions, shared by all the agents
  - generator of percepts
    - establishing which percepts for which agents

ENVIRONMENT MODELS IN MAS

- Environment as monolithic / centralised block
  - defining agent (external) actions
    - typically a static list of actions, shared by all the agents
  - generator of percepts
    - establishing which percepts for which agents
- Typically enough for ‘toy problems’, but...

ENVIRONMENT MODELS IN MAS

- Environment as monolithic / centralised block
  - defining agent (external) actions
    - typically a static list of actions, shared by all the agents
  - generator of percepts
    - establishing which percepts for which agents
- Typically enough for ‘toy problems’, but...
- No specific (programming) model for defining structure and behaviour
  - including concurrency management
    - relying on lower-level language features and mechanisms (i.e. Java)

ENVIRONMENT MODELS IN MAS

- Environment as monolithic / centralised block
  - defining agent (external) actions
    - typically a static list of actions, shared by all the agents
  - generator of percepts
    - establishing which percepts for which agents
- Typically enough for ‘toy problems’, but...
- No specific (programming) model for defining structure and behaviour
  - including concurrency management
    - relying on lower-level language features and mechanisms (i.e. Java)
- No support for interoperability across platforms

AGENTS AND PERCEPTION

AGENTS AND PERCEPTION

- Perception mechanisms in agents are often *built upon* the environment model
  - inheriting its weakness...
    - Computational models are often domain dependent
    - Based on the mechanisms inherited from OO (Java)

AGENTS AND PERCEPTION

- Perception mechanisms in agents are often *built upon* the environment model
  - inheriting its weakness...
    - Computational models are often domain dependent
    - Based on the mechanisms inherited from OO (Java)
- MAS: Great attention for message exchange and ACL
  - Sophisticated infrastructures for message passing, *but....*
  - Protocols, Ontologies = Computational load on Agents

AGENTS AND PERCEPTION

- Perception mechanisms in agents are often *built upon* the environment model
  - inheriting its weakness...
    - Computational models are often domain dependent
    - Based on the mechanisms inherited from OO (Java)
- MAS: Great attention for message exchange and ACL
  - Sophisticated infrastructures for message passing, *but....*
  - Protocols, Ontologies = Computational load on Agents
- In general, for other kinds of percepts:
  - Information flow is shaped based on the format provided by the adopted environment
  - No clear distinction, at a reasoning level, between practical and perceptive activities

JASON ENVIRONMENT

- GOLD-MINER DEMO -

```
public class MiningPlanet extends jason.environment.Environment {
  ...
  public void init(String[] args) {...}

  public boolean executeAction(String ag, Structure action) {
      boolean result = false;
      int agId = getAgIdBasedOnName(ag);
      if (action.equals(up)) {
        result = model.move(Move.UP, agId);
      } else if (action.equals(down)) {
        result = model.move(Move.DOWN, agId);
      } else if (action.equals(right)) {
        ...
      }
      return result;
  }

  private void updateAgPercept(String agName, int ag) {clearPercepts(agName);
    // its location
    Location l = model.getAgPos(ag);
    addPercept(agName, Literal.parseLiteral("pos(" + l.x + "," + l.y + ")"));
    if (model.isCarryingGold(ag)) {
      addPercept(agName, Literal.parseLiteral("carrying_gold"));
    }
    // what's around
    updateAgPercept(agName, l.x - 1, l.y - 1);
    updateAgPercept(agName, l.x - 1, l.y);
    ...
  }
}
```

ENRICHING THE VIEW: WORK ENVIRONMENTS

ENRICHING THE VIEW: WORK ENVIRONMENTS

- Perspective: *designing worlds in agent systems* for agents' *activities*
  - designing good and effective place for agents to live and work in
    - environment as the context of agent activities *inside the MAS*
  - beyond simulated worlds

ENRICHING THE VIEW: WORK ENVIRONMENTS

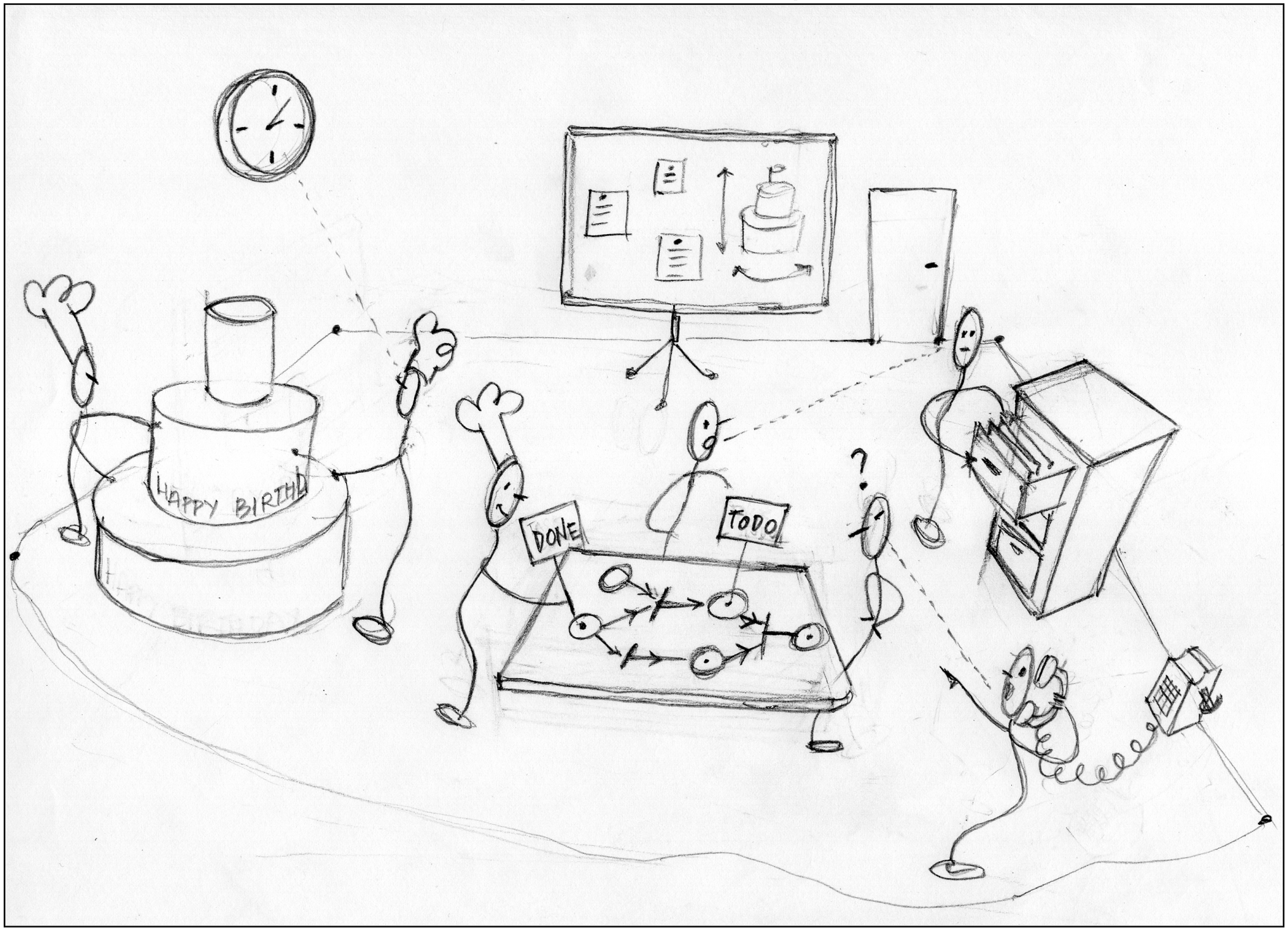
- Perspective: *designing worlds in agent systems* for agents' *activities*
  - designing good and effective place for agents to live and work in
    - environment as the context of agent activities *inside the MAS*
  - beyond simulated worlds
- "**Work environment**" notion
  - that *part of the MAS* that is *designed* and *programmed* so as to ease agent activities and work
    - first-class entity of the agent world

ENRICHING THE VIEW: WORK ENVIRONMENTS

- Perspective: *designing worlds in agent systems* for agents' *activities*
  - designing good and effective place for agents to live and work in
    - environment as the context of agent activities *inside the MAS*
  - beyond simulated worlds
- "**Work environment**" notion
  - that *part of the MAS* that is *designed* and *programmed* so as to ease agent activities and work
    - first-class entity of the agent world
- as pivotal element of MAS design and programming

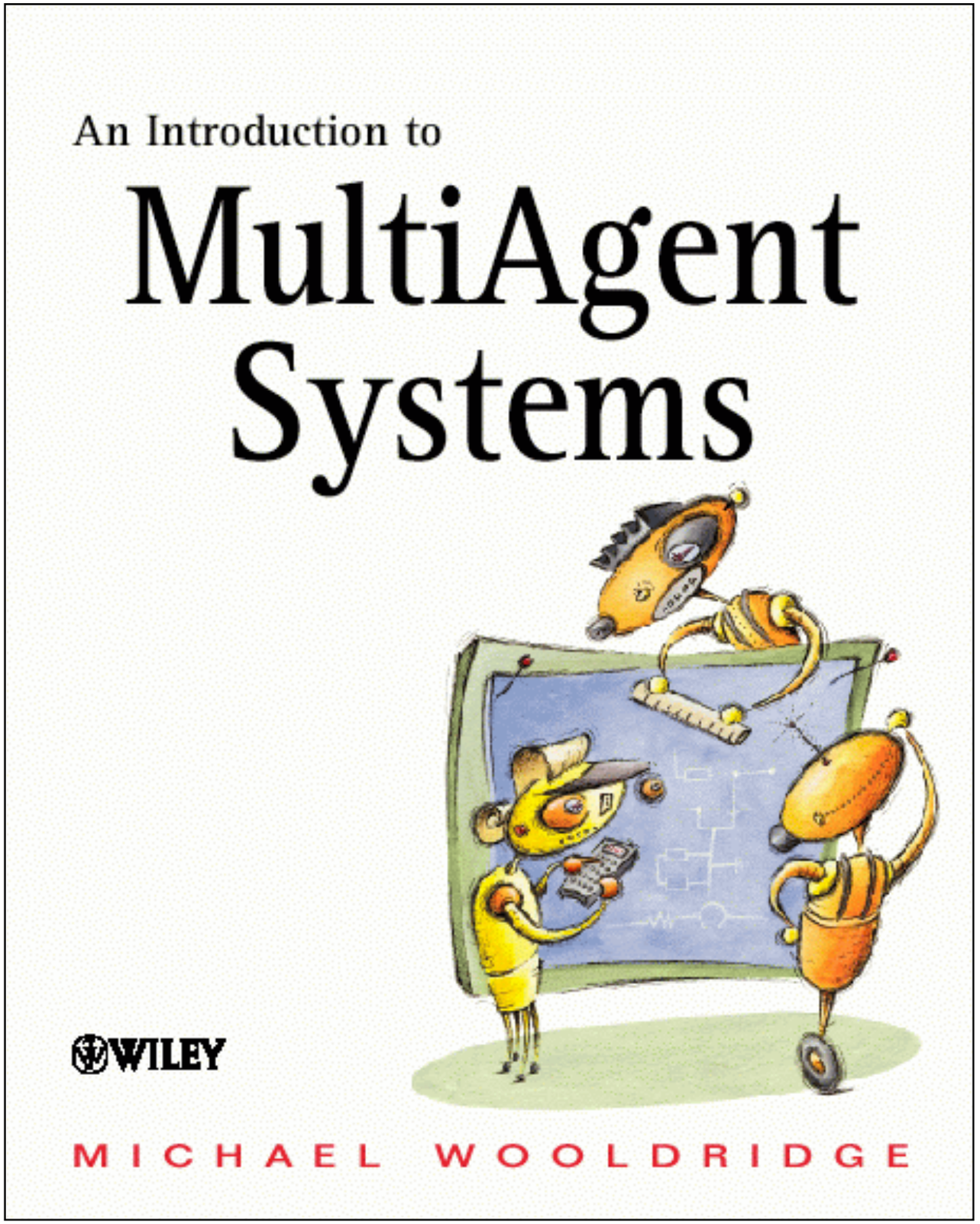
A HUMAN WORK ENVIRONMENT

(~BAKERY)



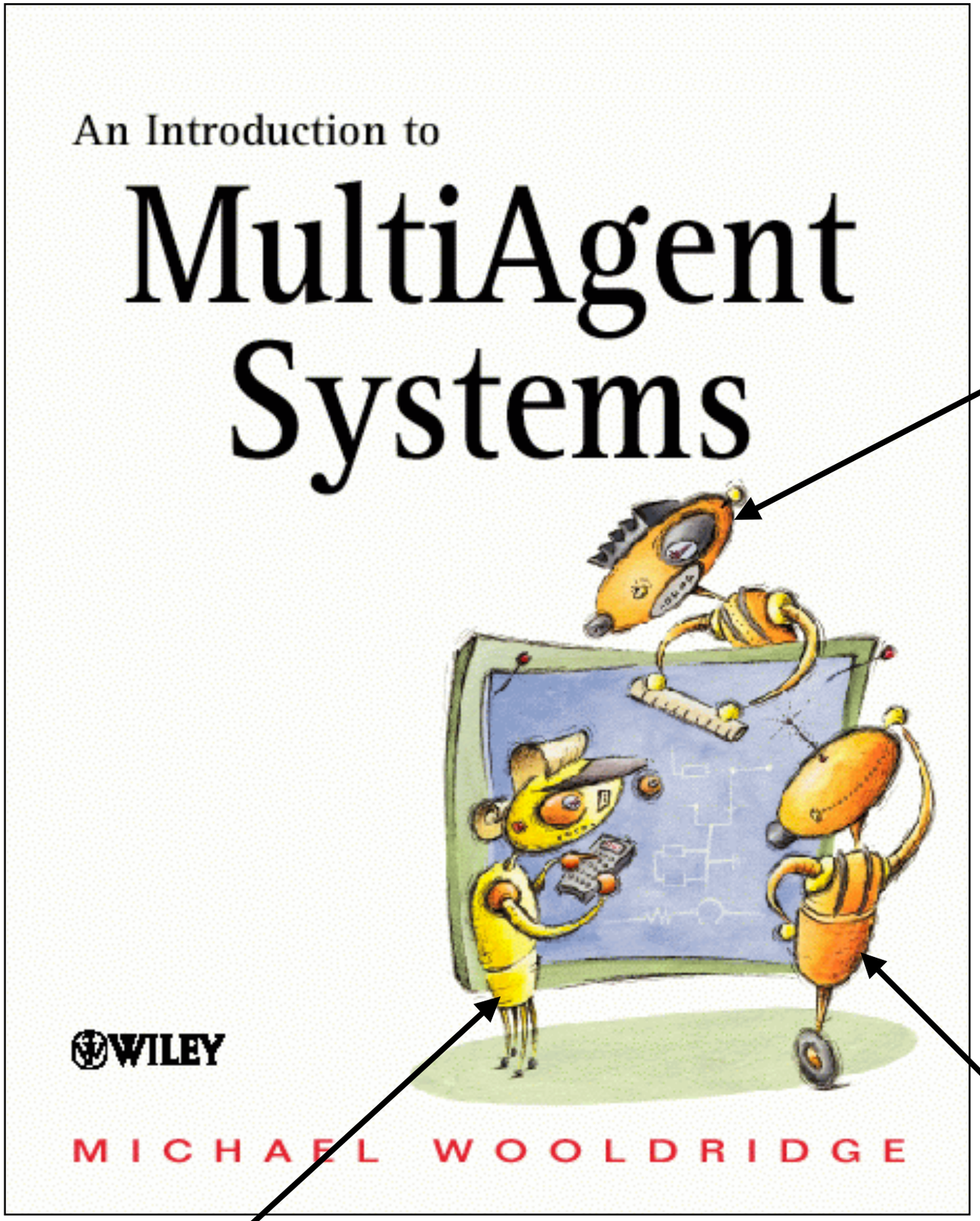
ARTIFACTS ARE IN THE AGENT MAINSTREAM

...not really, actually...



ARTIFACTS ARE IN THE AGENT MAINSTREAM

...not really, actually...



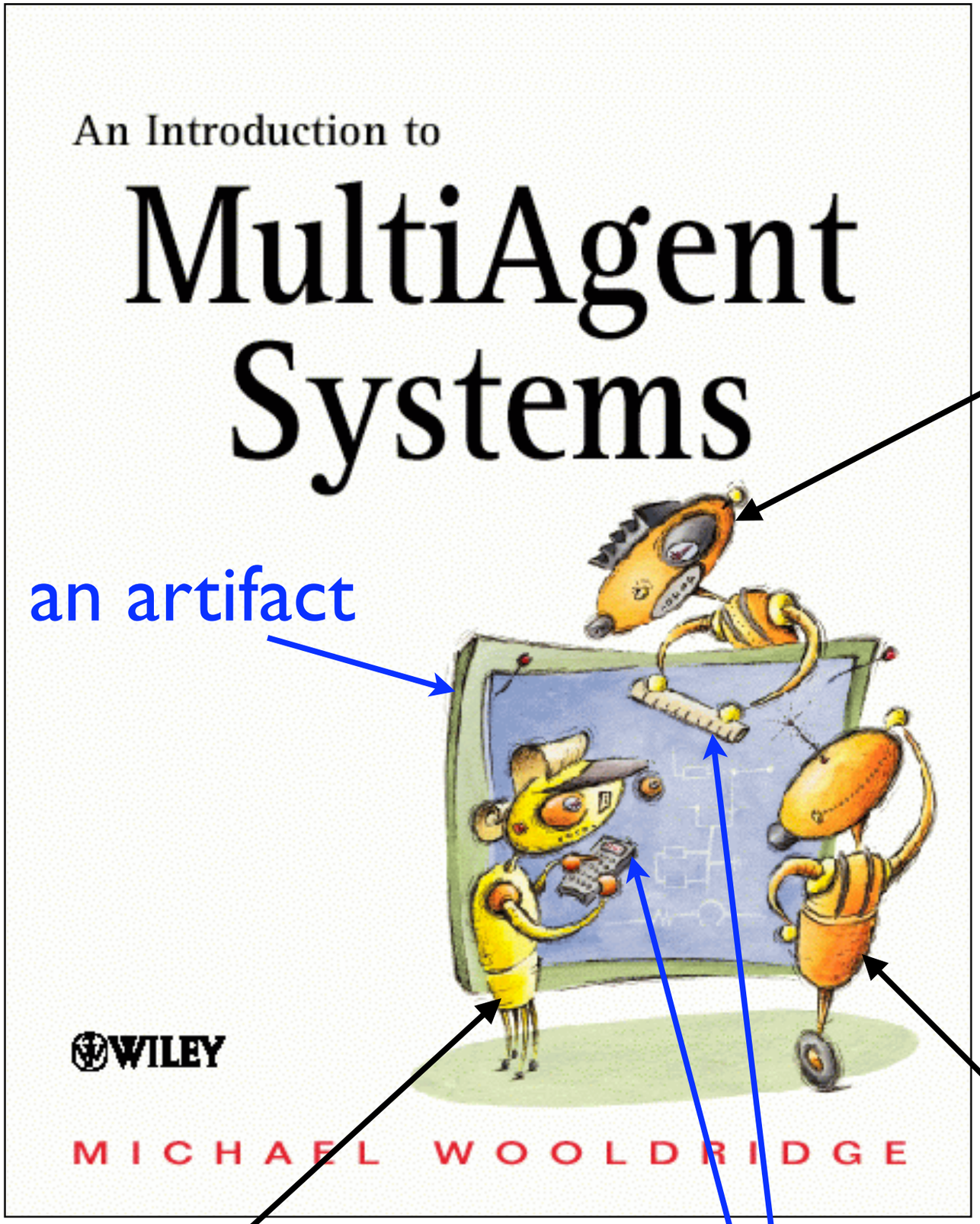
agent

agent

agent

ARTIFACTS ARE IN THE AGENT MAINSTREAM

...not really, actually...



agent

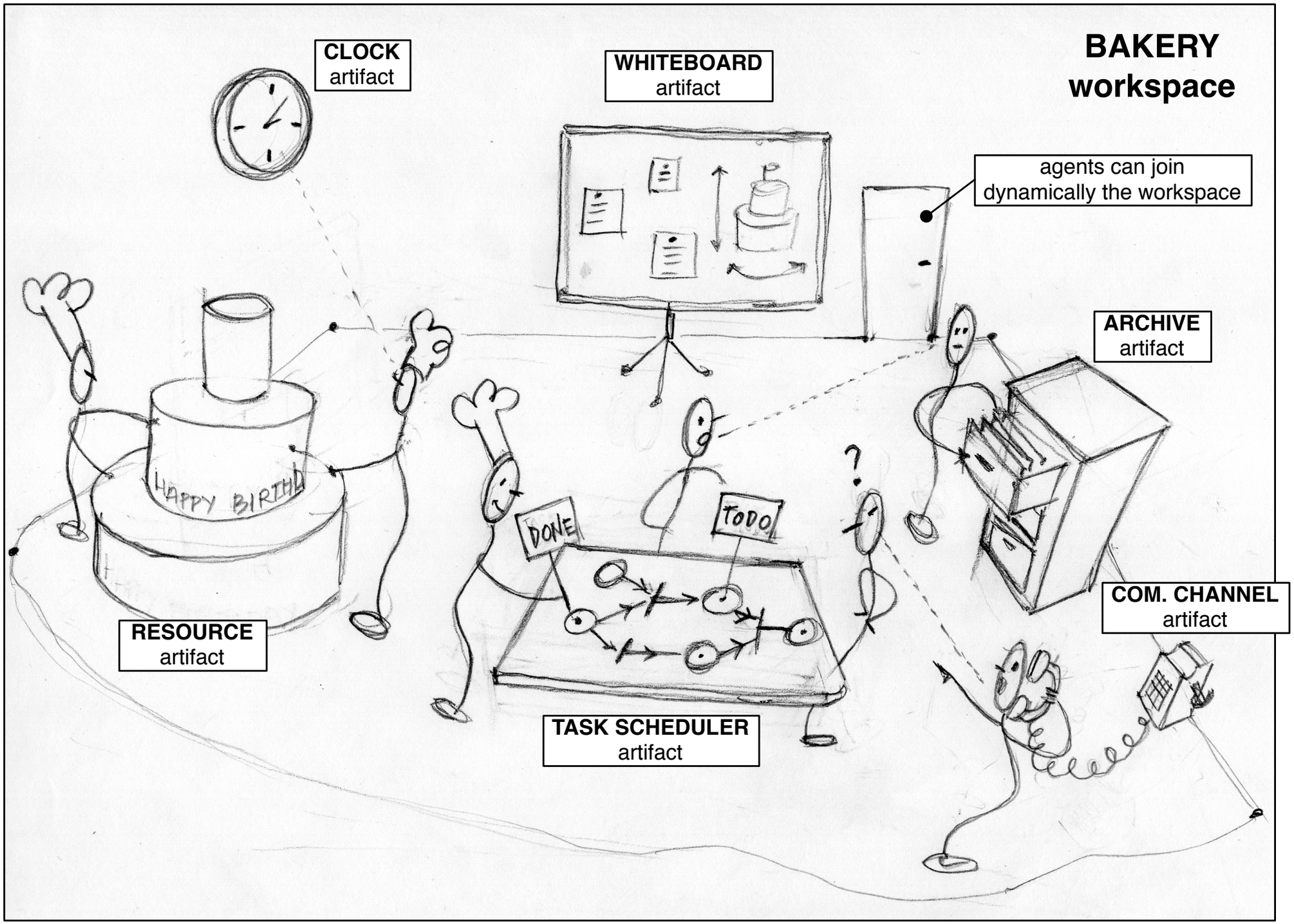
an artifact

agent

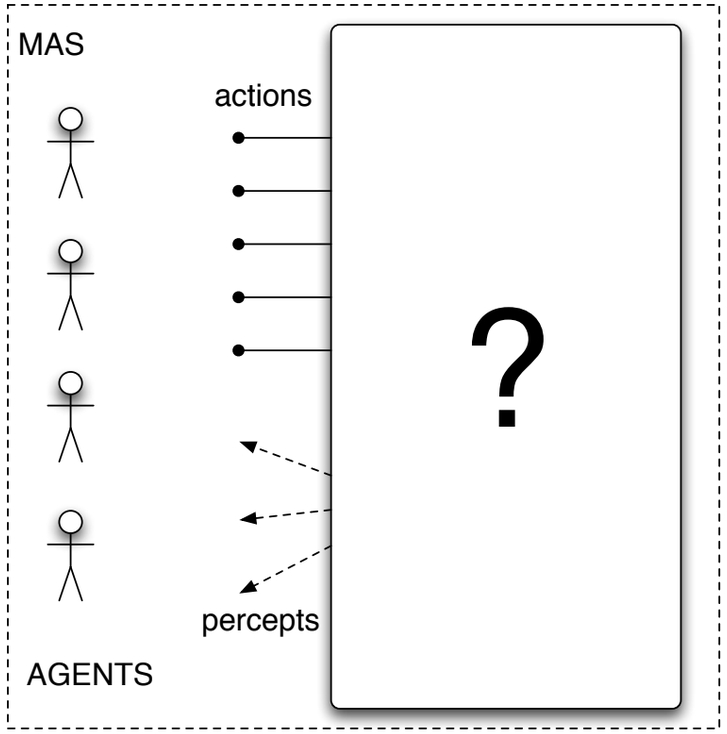
artifacts

agent

AGENTS & ARTIFACTS (A&A) MODEL: BASIC IDEA IN A PICTURE

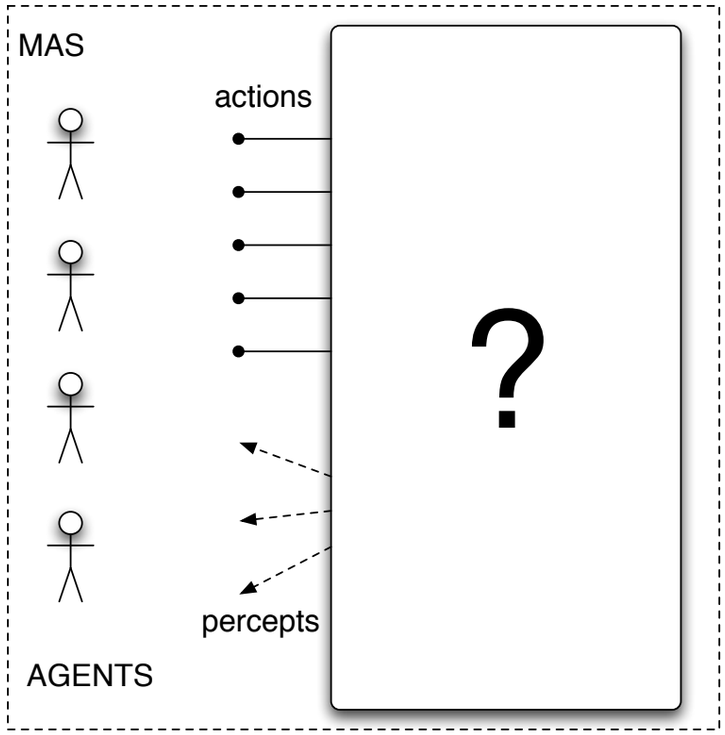


DESIDERATA FOR A WORK. ENV. PROGRAMMING MODEL (1/2)



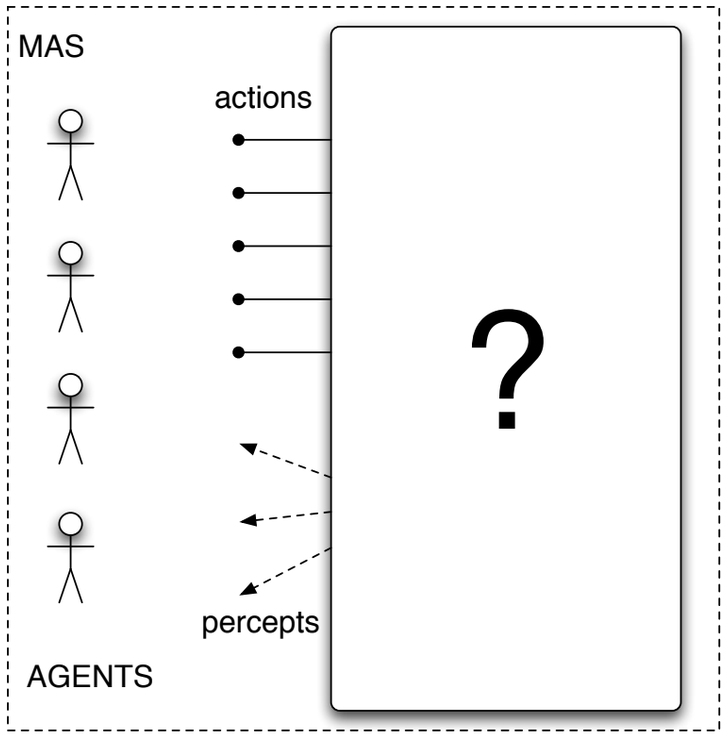
DESIDERATA FOR A WORK. ENV. PROGRAMMING MODEL (1/2)

- **Abstraction**
  - keeping the agent abstraction level
    - concepts like Action, Perception, Observation
    - i.e., no agents sharing and calling *OO objects*
  - effective programming models
    - for controllable and observable computational entities



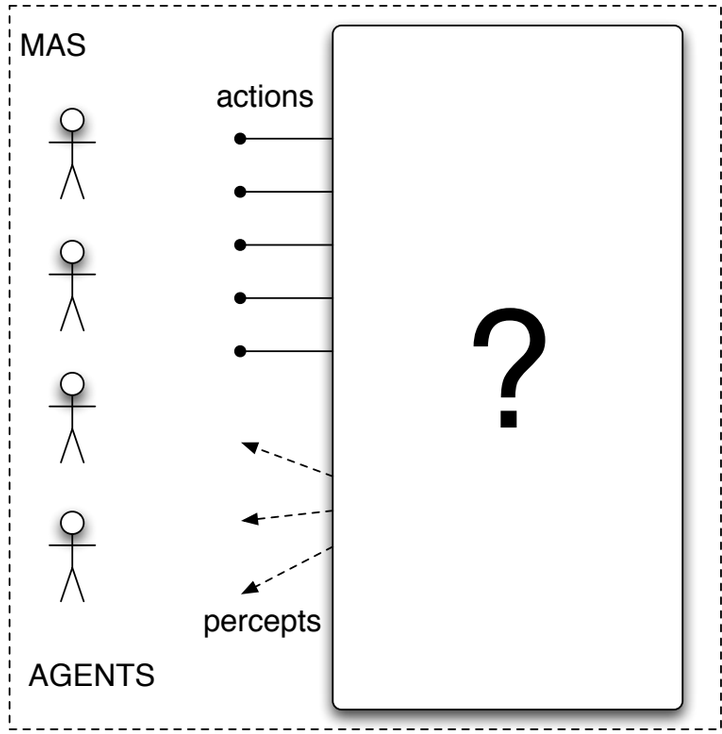
DESIDERATA FOR A WORK. ENV. PROGRAMMING MODEL (1/2)

- **Abstraction**
  - keeping the agent abstraction level
    - concepts like Action, Perception, Observation
    - i.e., no agents sharing and calling *OO objects*
  - effective programming models
    - for controllable and observable computational entities



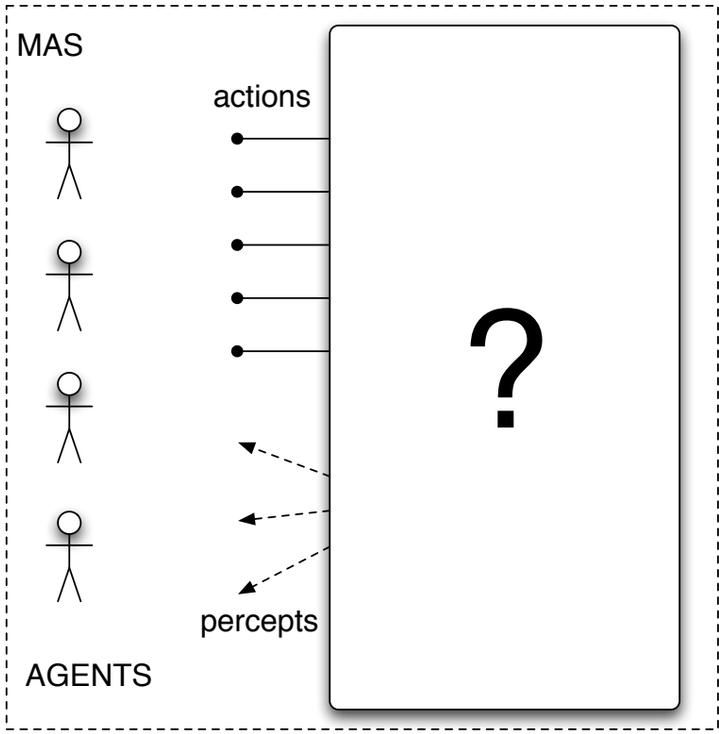
- **Orthogonality**
  - wrt agent models, architectures, platforms
  - support for heterogeneous systems

DESIDERATA FOR A WORK ENV. PROGRAMMING MODEL (2/2)



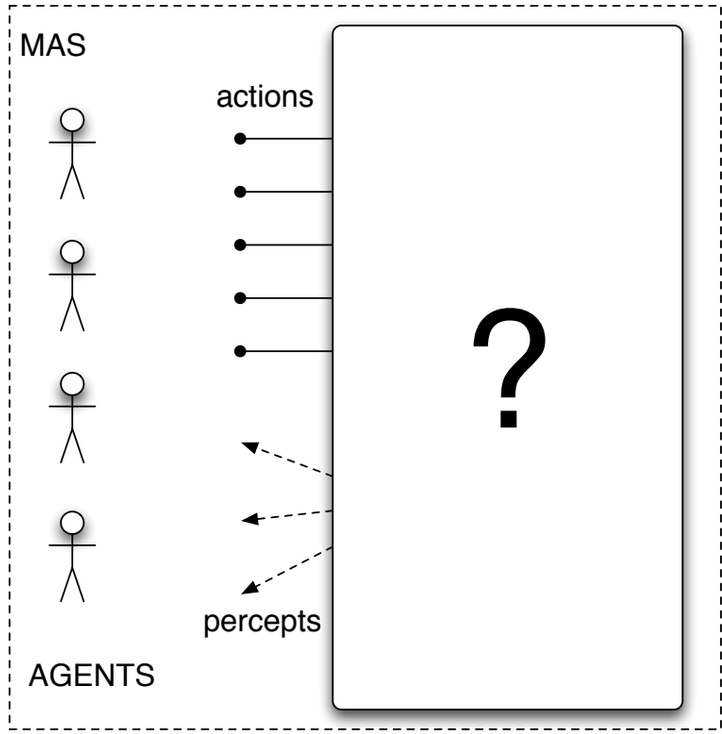
DESIDERATA FOR A WORK ENV. PROGRAMMING MODEL (2/2)

- **(Dynamic) extendibility**
  - dynamic construction, replacement, extension of environment parts
  - support for *open* systems



DESIDERATA FOR A WORK ENV. PROGRAMMING MODEL (2/2)

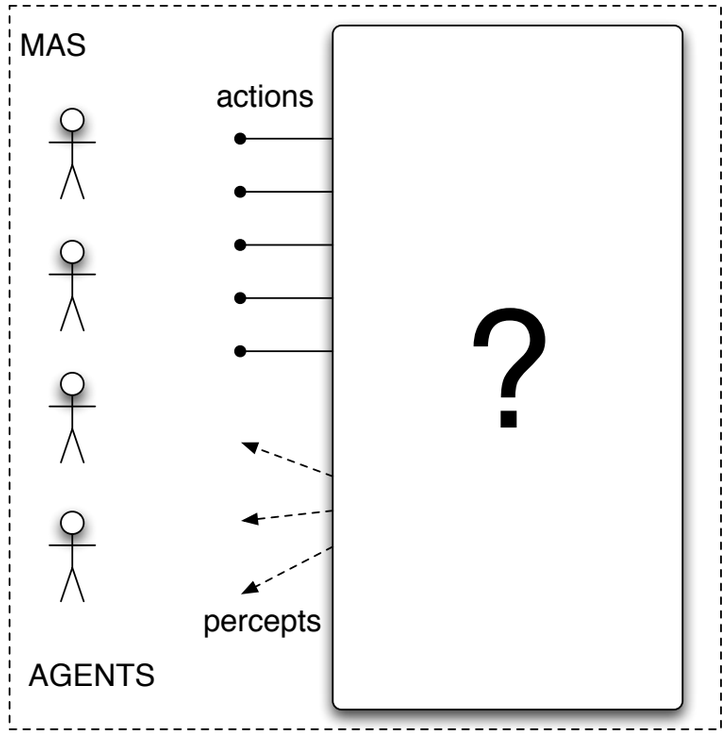
- **(Dynamic) extendibility**
  - dynamic construction, replacement, extension of environment parts
  - support for *open* systems



- **Modularity**
  - away from the monolithic and centralised view

DESIDERATA FOR A WORK ENV. PROGRAMMING MODEL (2/2)

- **(Dynamic) extendibility**
  - dynamic construction, replacement, extension of environment parts
  - support for *open* systems



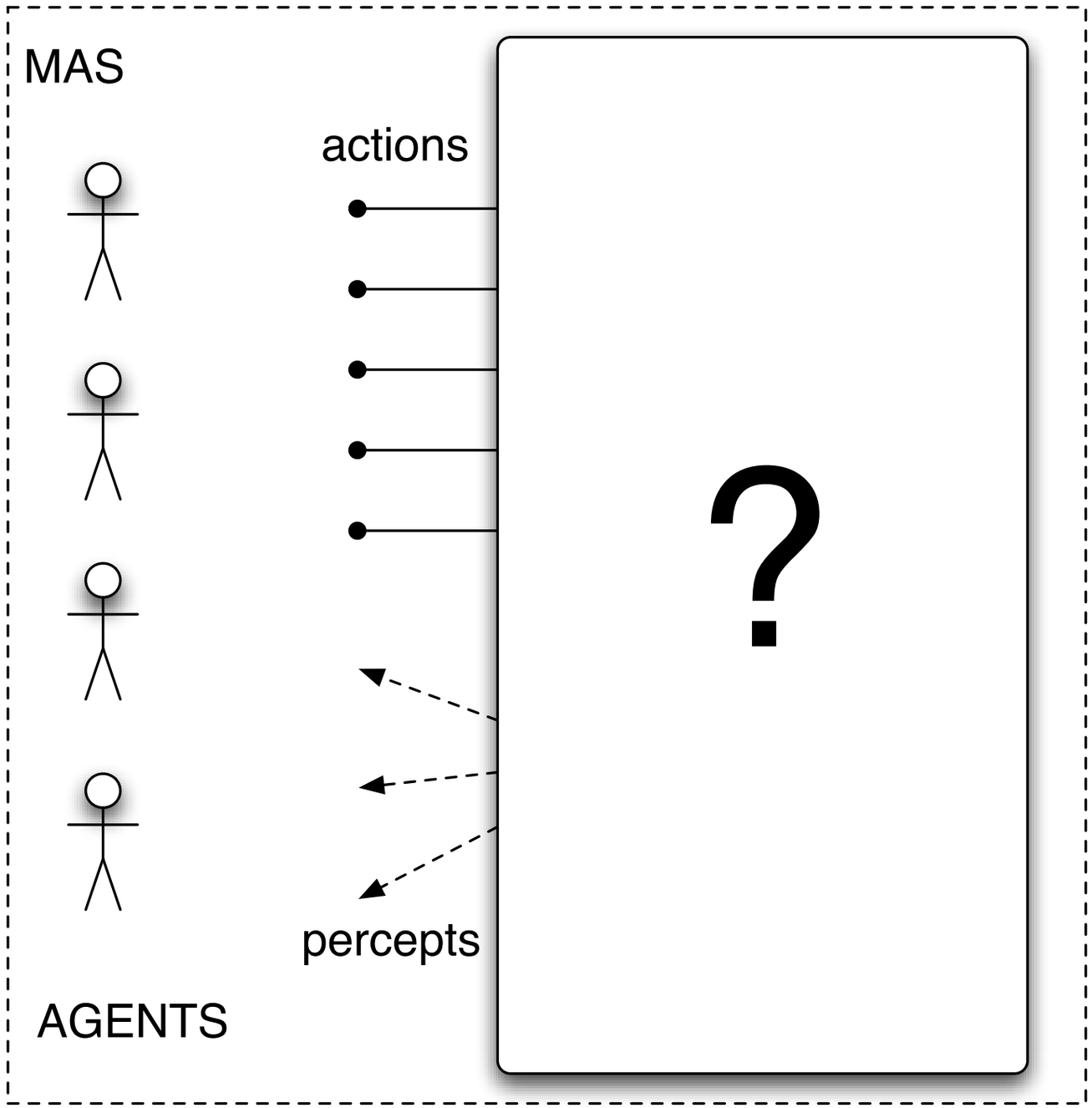
- **Modularity**
  - away from the monolithic and centralised view

- **Reusability**
  - reuse of environment parts in different application contexts / domains

A&A BASIC CONCEPTS

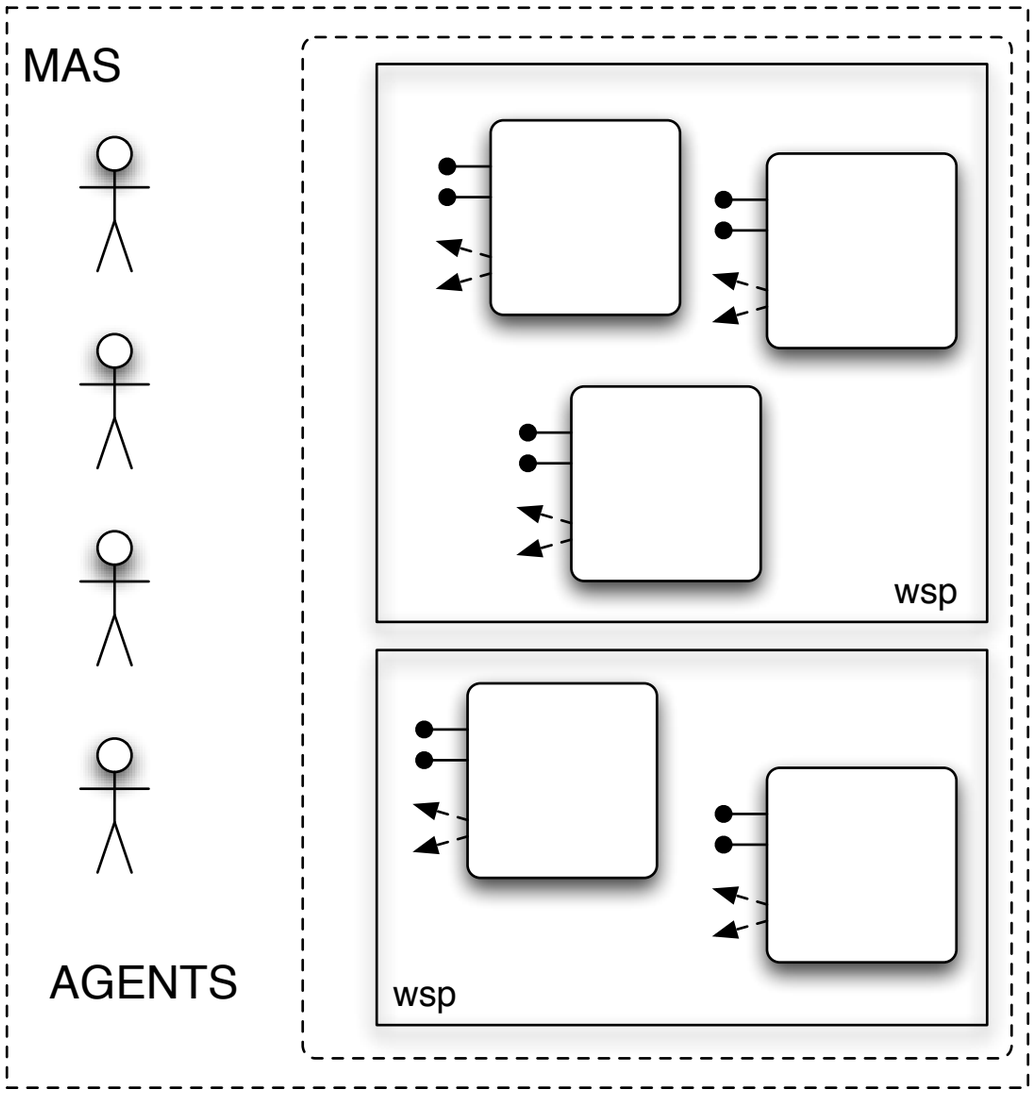
- **Agents**
  - autonomous, goal-oriented, social, pro-active entities
  - create and co-use artifacts for supporting their activities
    - besides direct communication
- **Artifacts**
  - *non-autonomous*, *authomatic, function*-oriented entities
    - controllable and observable
  - modelling the tools and resources used by agents
    - designed by MAS programmers
- **Workspaces**
  - logically grouping agents & artifacts
  - defining the topology of the computational environment

WORK ENVIRONMENT IN A&A



WORK ENVIRONMENT IN A&A

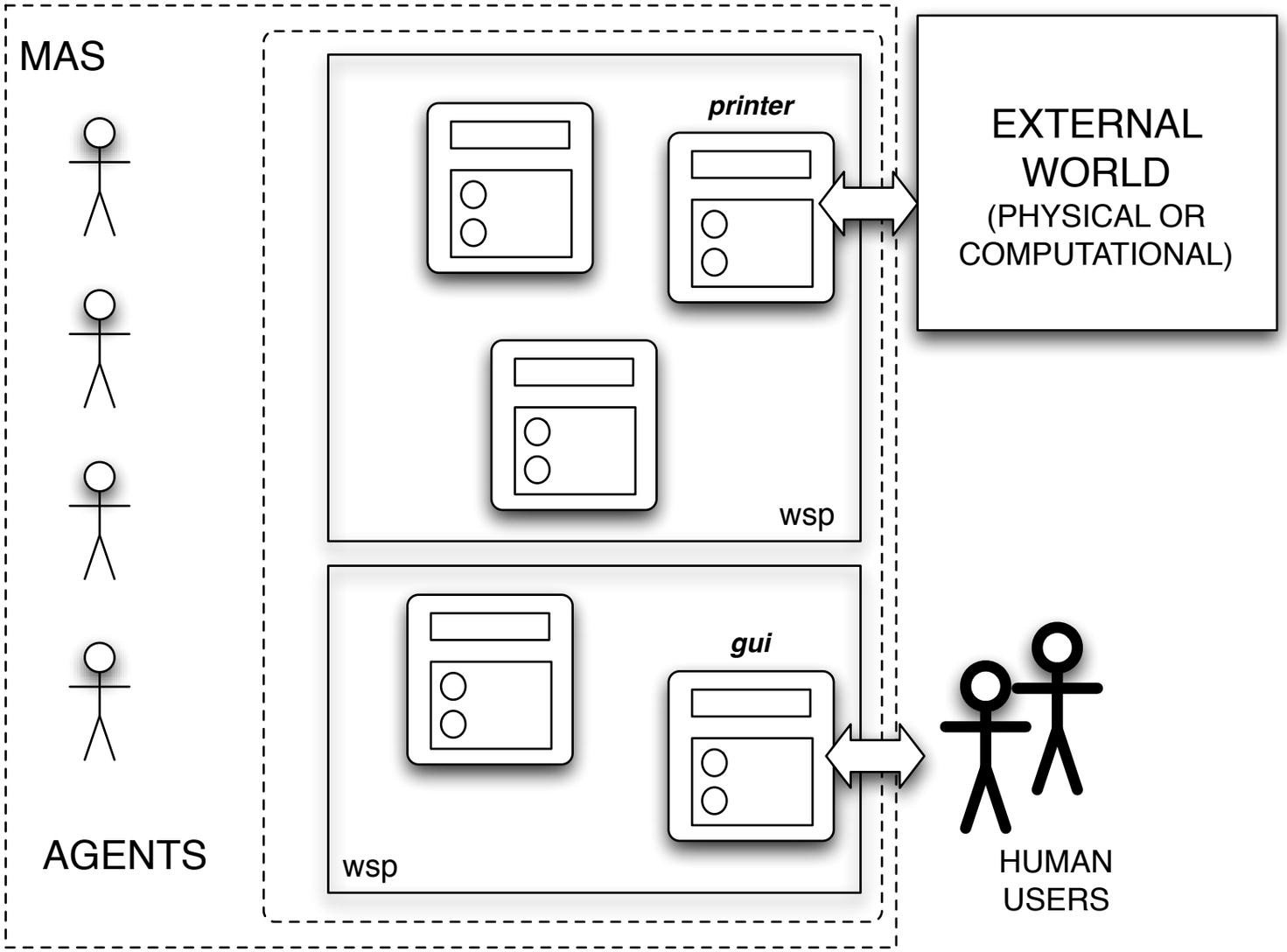
- Artifacts as building bricks to structure the (work) environment
- Workspaces as topological structure of the w.e.



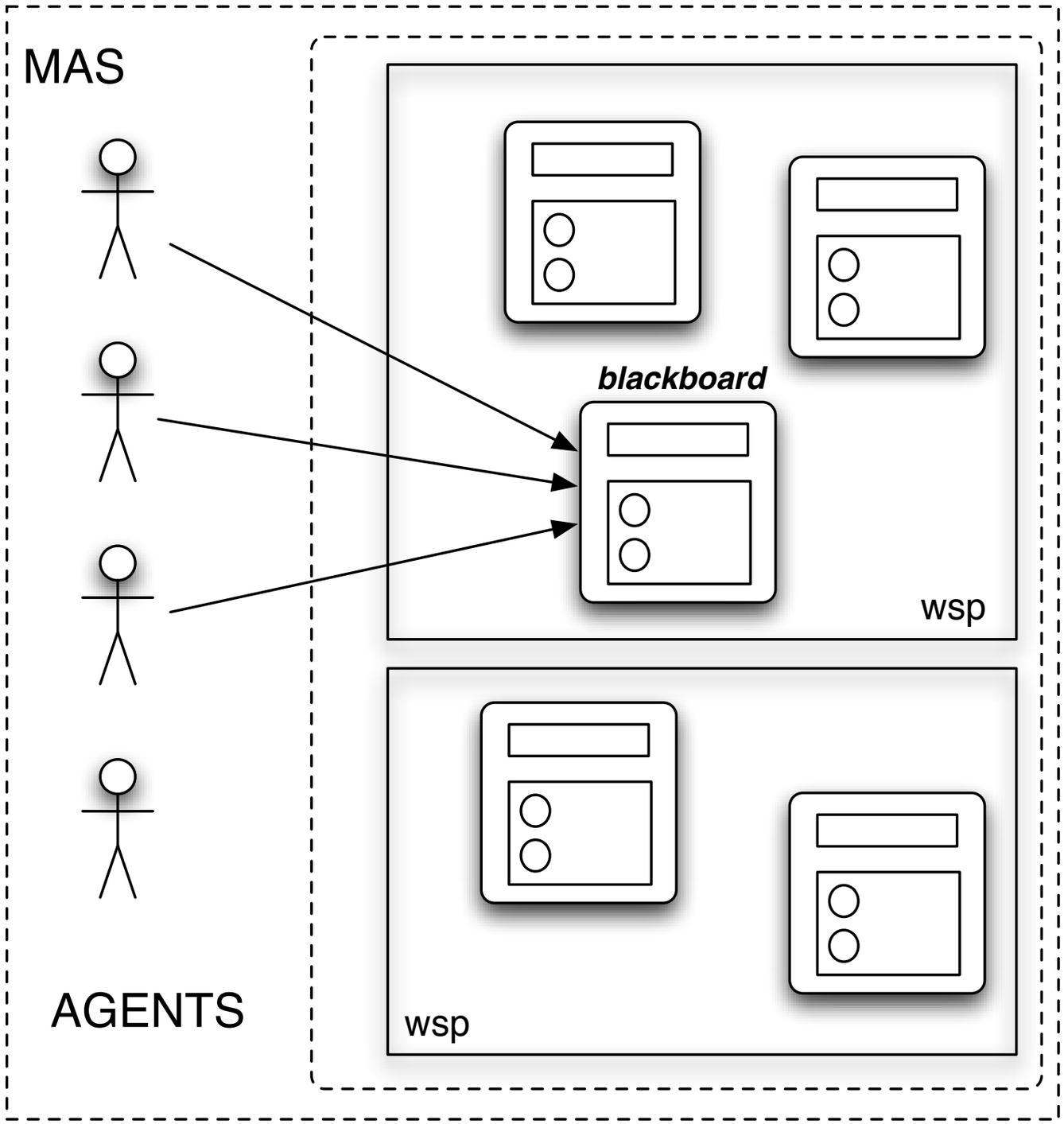
- Artifacts as Special Purpose computational ‘objects’
- Agents and Artifacts are not alternative notions, are *dual* notions

WORK ENVIRONMENT IN A&A

- Artifacts as Interfaces and Facades to external world(s)



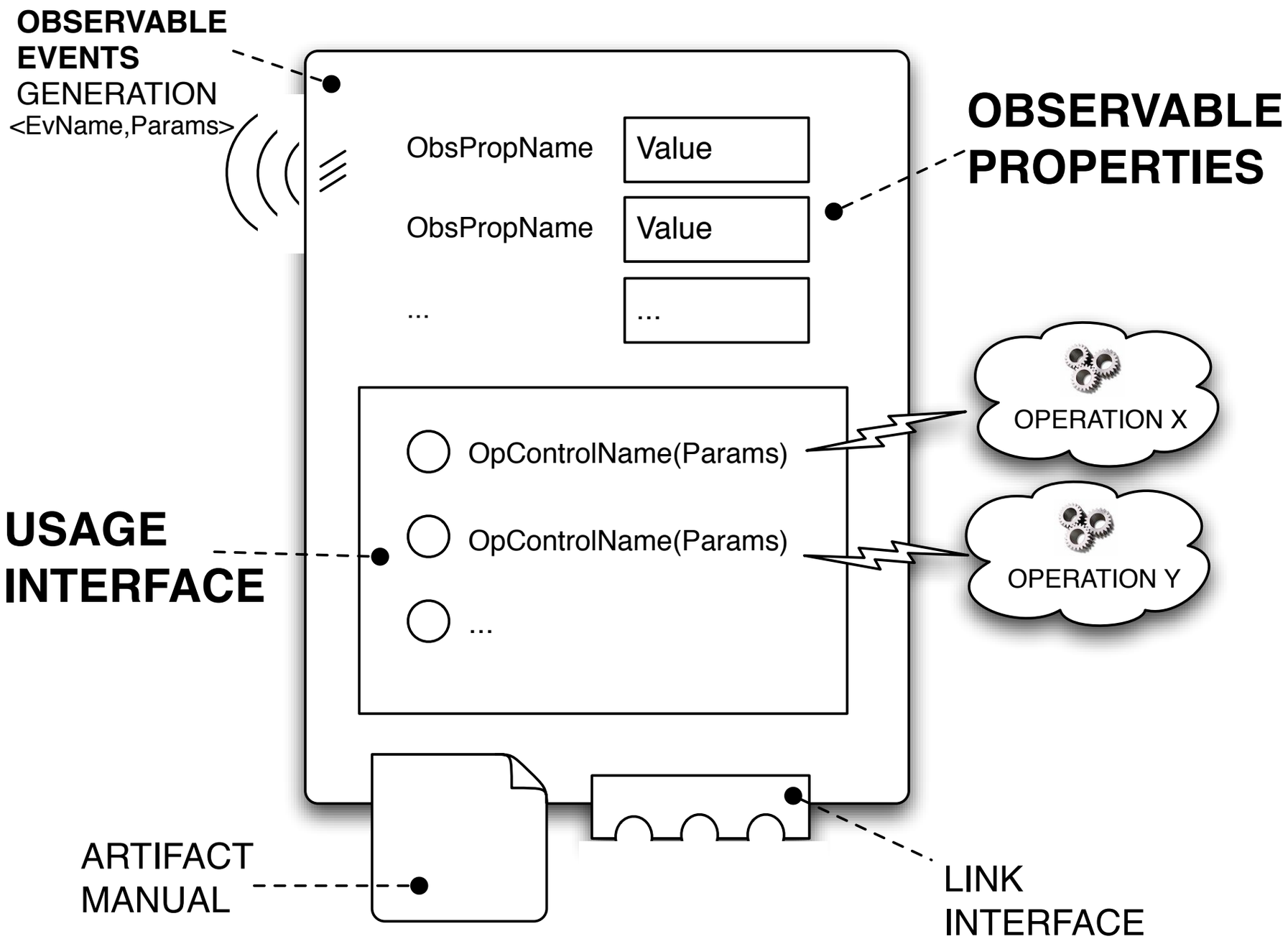
WORK ENVIRONMENT IN A&A



- Interaction of Ag and Art based on not ambigous notions of
  - Action
  - Perception
  - Observation

ARTIFACT COMPUTATIONAL MODEL

- “COFFEE MACHINE METAPHOR” -



CArtAgO

CArtAgO

- **CArtAgO** platform / infrastructure
  - runtime environment for executing (possibly distributed) artifact-based environments
  - Java-based programming model for defining artifacts
  - set of basic AGENT-API to work wihin artifact-based environment
    - integration with agent programming platforms

CArtAgO

- **CArtAgO** platform / infrastructure
  - runtime environment for executing (possibly distributed) artifact-based environments
  - Java-based programming model for defining artifacts
  - set of basic AGENT-API to work wihin artifact-based environment
    - integration with agent programming platforms
- Distributed and open MAS
  - workspaces distributed on Internet nodes
    - agents can join and work in multiple workspace at a time
  - Role-Based Access Control (RBAC) security model

CArtAgO

- **CArtAgO** platform / infrastructure
  - runtime environment for executing (possibly distributed) artifact-based environments
  - Java-based programming model for defining artifacts
  - set of basic AGENT-API to work wihin artifact-based environment
    - integration with agent programming platforms
- Distributed and open MAS
  - workspaces distributed on Internet nodes
    - agents can join and work in multiple workspace at a time
  - Role-Based Access Control (RBAC) security model
- Open-source technology
  - available at http://cartago.sourceforge.net

...AND FRIENDS

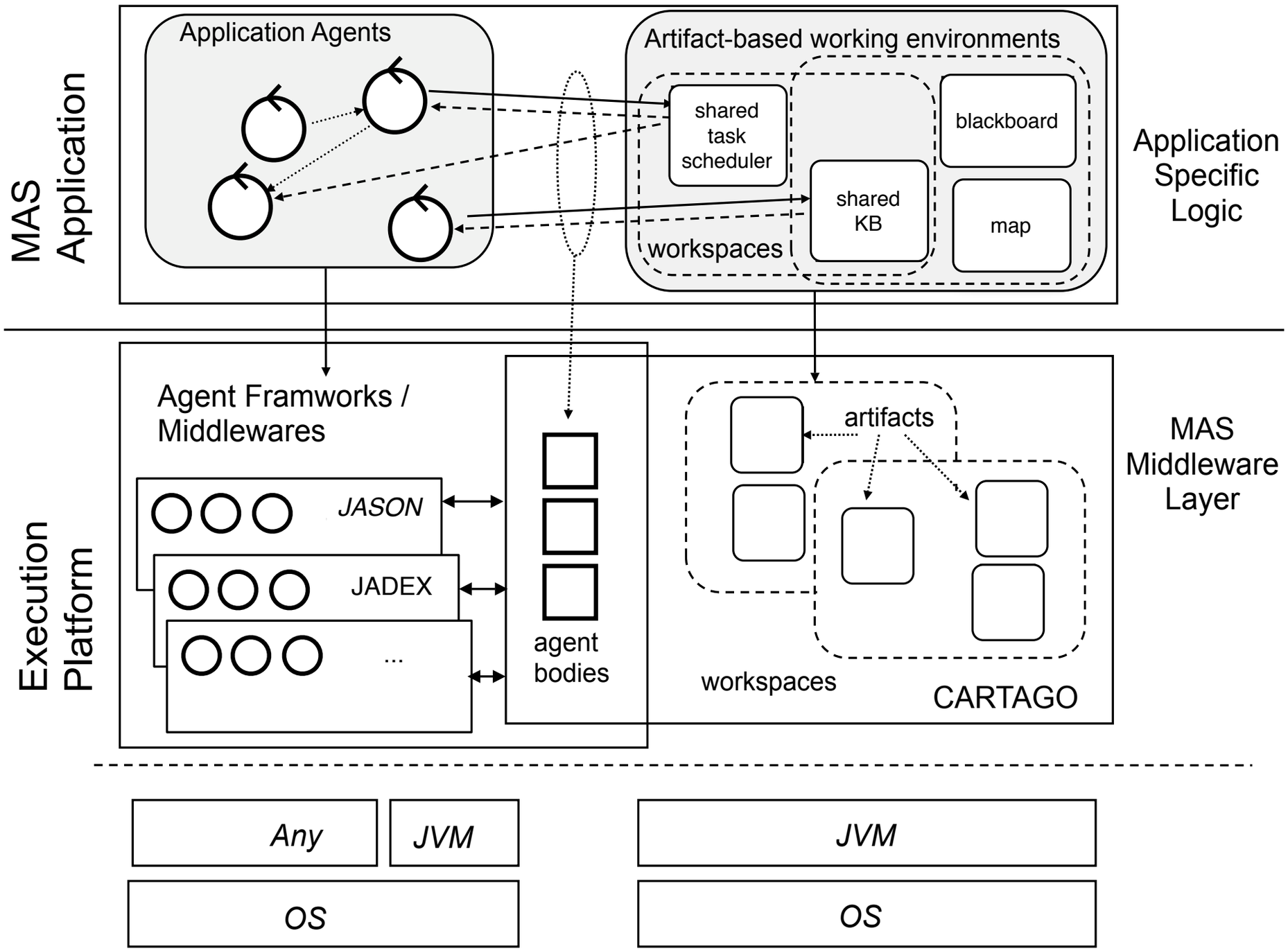
…AND FRIENDS

- Integration with existing agent platforms
  - cognitive agent platforms in particular
    - ongoing cooperation with Jomi, Rafael, Alexander, Lars, Mehdi
  - available bridges: ***Jason***, **Jadex**, simpA
    - ongoing: **2APL**, Jade
  - “agent body” notion for technically realising the integration
    - effectors and sensors to act upon and sense artifacts
    - controlled by an agent mind executed on some agent platform

…AND FRIENDS

- Integration with existing agent platforms
  - cognitive agent platforms in particular
    - ongoing cooperation with Jomi, Rafael, Alexander, Lars, Mehdi
  - available bridges: ***Jason***, **Jadex**, simpA
    - ongoing: **2APL**, Jade
  - "agent body" notion for technically realising the integration
    - effectors and sensors to act upon and sense artifacts
    - controlled by an agent mind executed on some agent platform
- Outcome
  - developing open and heterogenous MAS
  - different perspective on *interoperability*
    - sharing and working in a common work environment
    - common data-model based on Object-Oriented or XML-based data structures

CArtAgO ARCHITECTURE



INTEGRATING HETEROGENEOUS AGENTS PLATFORMS

Extending agents repertoire with a basic set of actions enabling playing within artifact-based environment

workspace management	**joinWsp**(Name,?WspId,+Node,+Role,+Cred) **quitWsp**(Wid)
artifact use	**use**(Aid,OpCntrName(Params),+Sensor,+Timeout,+Filter) **sense**(Sensor,?Perception,+Filter,+Timeout) **grab**([Aid]) **release**([Aid])
artifact observation	**observeProperty**(Aid,PName,?PValue) **focus**(Aid,+Sensor,+Filter) **stopFocus**(Aid)
artifact instantiation, discovery, management	**makeArtifact**(Name,Template,+ArtifactConfig,?Aid) **lookupArtifact**(Name,?Aid) **disposeArtifact**(Aid)

Conceiving A&A Systems

So: artifacts provide serviceable operations, preprocessed information and coordination facilities conceived by the MAS designer for easing agents tasks

Conceiving A&A Systems

So: artifacts provide serviceable operations, preprocessed information and coordination facilities conceived by the MAS designer for easing agents tasks

- What's for agent architectures and systems...
  - ...When Agents are reasoning? /w mental states (Beliefs, Goals, Intentions)?

Conceiving A&A Systems

So: artifacts provide serviceable operations, preprocessed information and coordination facilities conceived by the MAS designer for easing agents tasks

- What's for agent architectures and systems...
  - ...When Agents are reasoning? /w mental states (Beliefs, Goals, Intentions)?
- Functional Approach to A&A design

Conceiving A&A Systems

So: artifacts provide serviceable operations, preprocessed information and coordination facilities conceived by the MAS designer for easing agents tasks

- What's for agent architectures and systems...
  - ...When Agents are reasoning? /w mental states (Beliefs, Goals, Intentions)?
- Functional Approach to A&A design
- From an agent viewpoint, artifacts embeds:
  - **Informational (Doxastic) Function**
  - **Operational (Purposive) Function**

Operational Function

- Operations are Artifacts *intended* purposes
  - *"Intented"* is in the mind of Artifact Designers before beeing in agents intentions
- 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

Operational Function

- Goals can thus be achieved by the mean of operations which have been defined *by the artifact developer* within artifacts control interface.
- *Agent's Goal 'is' in Operation Outcomes*:

```
+!doAction
      : not done(myGoal) & mySensor(S) & artifactID(Aid)
      <- cartago.use(Aid, doAction(X, Y, ...), S);
         cartago.sense(S, done(Res) );
         +done(Res).  /* Res is myGoal */

+!doAction : true
      <- ?done(myGoal).
```

Doxastic 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 in environments (*situated cognition*)

Doxastic Function

- Artifact Observable properties can be exploited by agents to retrieve strategic information
- The mechanism of observation is conceived as a *weak* interaction
- *Strategic (Goal Supporting) knowledge 'is' in artifacts readable properties:*

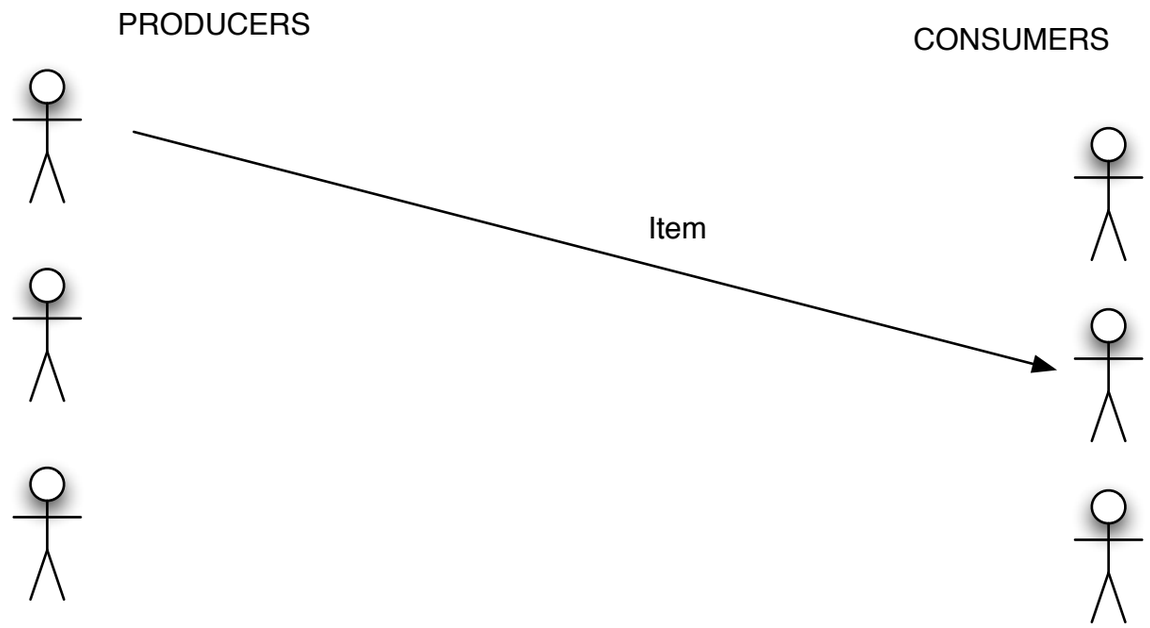
Doxastic Function

- Artifact Observable properties can be exploited by agents to retrieve strategic information
- The mechanism of observation is conceived as a *weak* interaction
- *Strategic (Goal Supporting) knowledge 'is' in artifacts readable properties:*

```
+!achieveGoal : artifactID(Aid)
  <-  cartago.observeProperty(Aid, propertyName(X) );
      !doAction(X).

+!doAction(X)    /* X is a Goal Supporting Belief */
  : X > 40
  <- +done(myGoal).

+!doAction(X)
  : X > 20
  <- cartago.use(Aid, doAction(X, Y), S);
     cartago.sense(S, done(Res) );
     +done(Res).  /* Res is myGoal */
```

PRODUCERS CONSUMERS

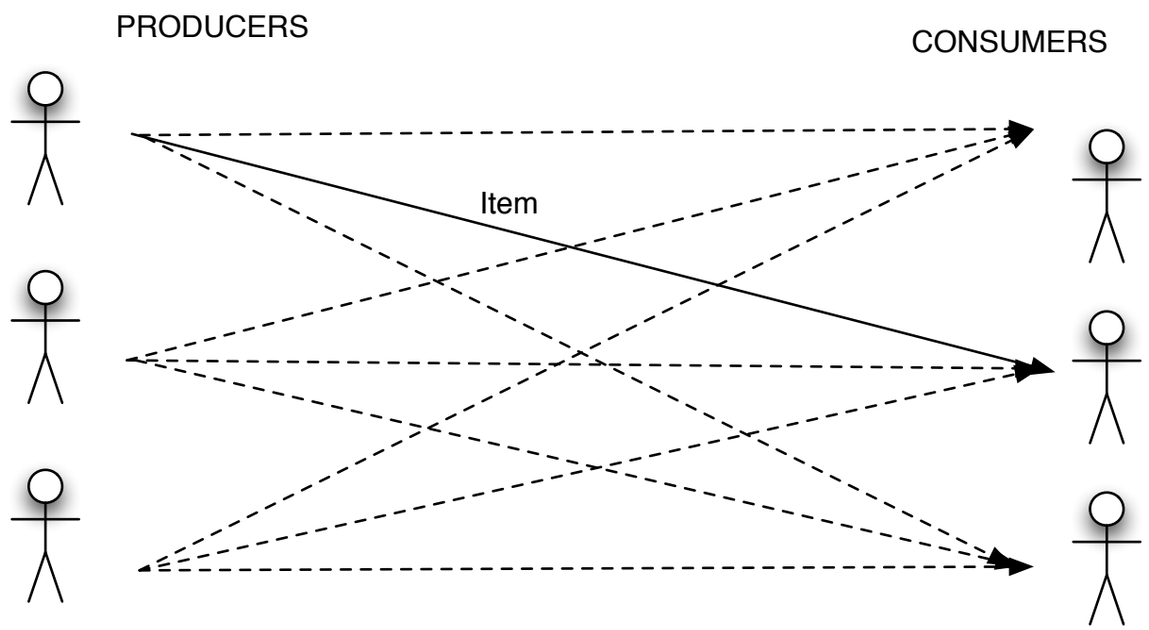
- Example: a variable number of Producer and Consumer Agents want to exchange generic Informational Items



- Agents have bounded resources: Time, Space, Memory…
  - Need for a strategy to synchronise / coordinate exchanges
  - Well known problem in concurrent systems
  - What's in MAS ?

PRODUCERS CONSUMERS IN MAS

- Direct Interaction: overload of messages and broadcasts (who may I ask for a relevant Item ?)

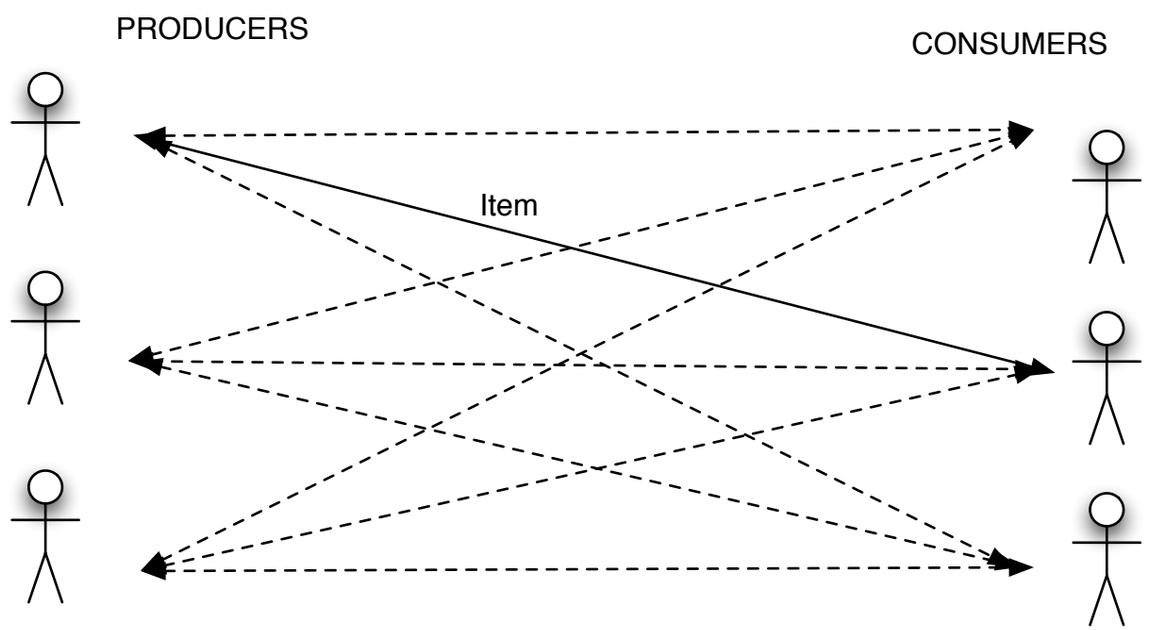


PRODUCERS CONSUMERS IN MAS

- Direct Interaction: overload of messages and broadcasts (who may I ask for a relevant Item ?)

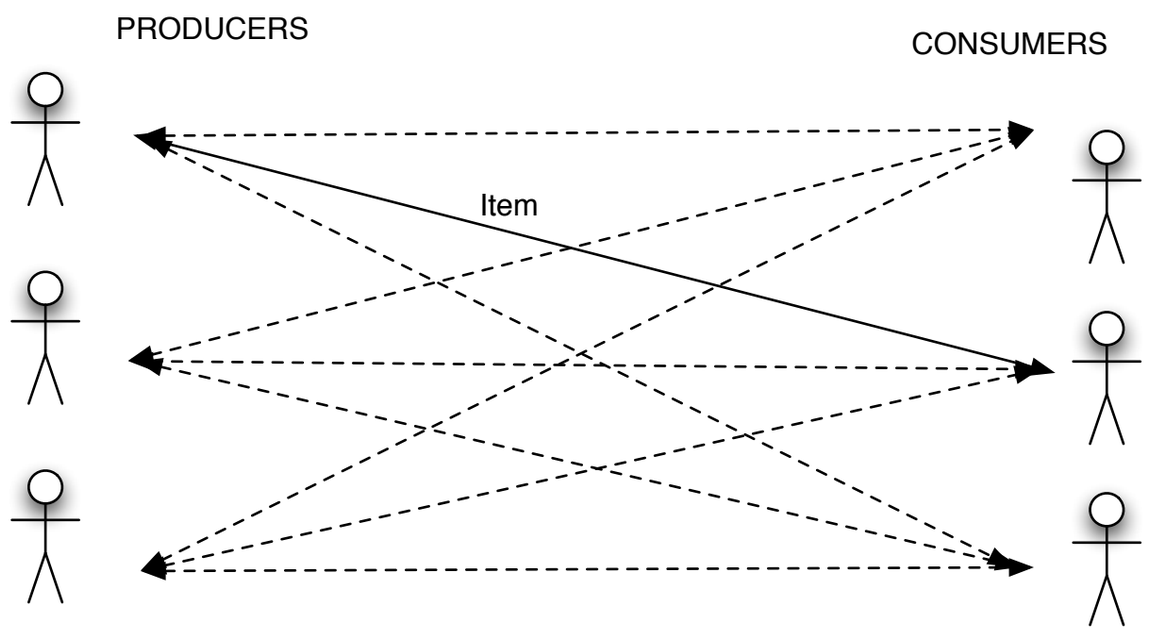
PRODUCERS CONSUMERS IN MAS

- Direct Interaction: overload of messages and broadcasts (who may I ask for a relevant Item ?)



PRODUCERS CONSUMERS IN MAS

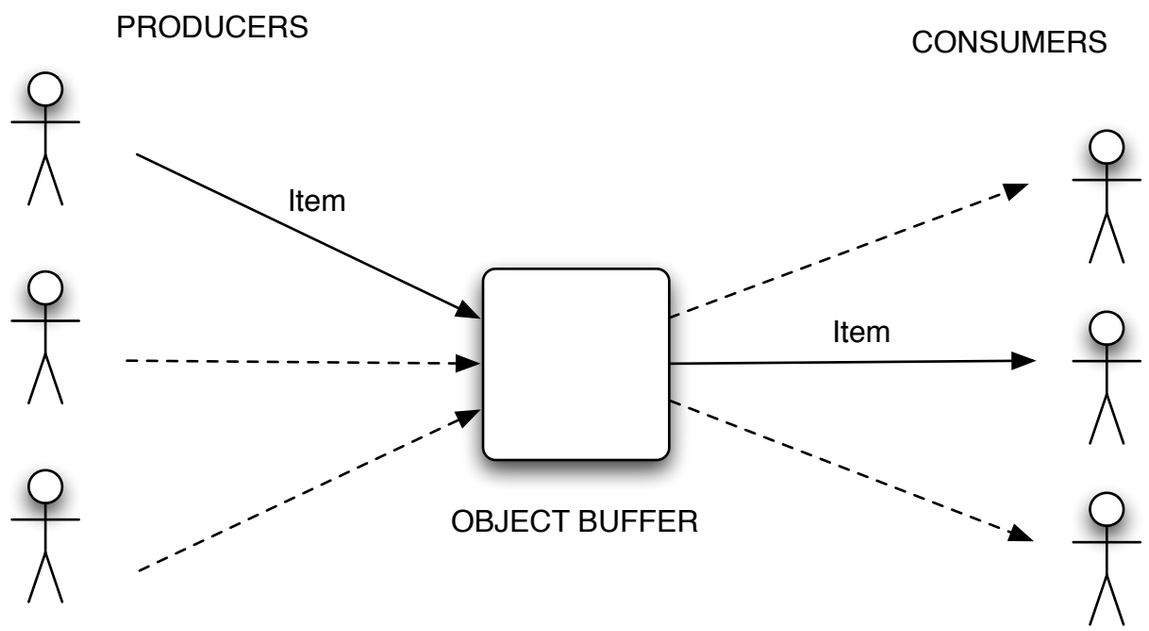
- Direct Interaction: overload of messages and broadcasts (who may I ask for a relevant Item ?)



- Which need to be typed (protocols)
- Bidirectional (Request-Response, Solicit Response...)

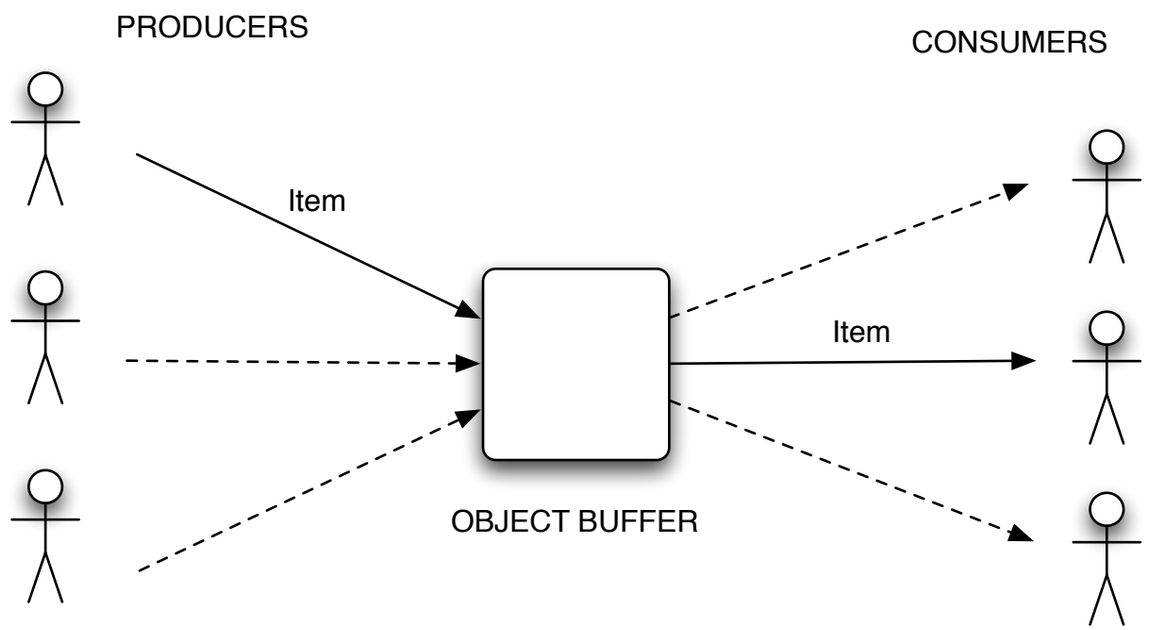
PRODUCERS CONSUMERS IN MAS

- Use an Object as a shared buffer reduces the number of interactions



PRODUCERS CONSUMERS IN MAS

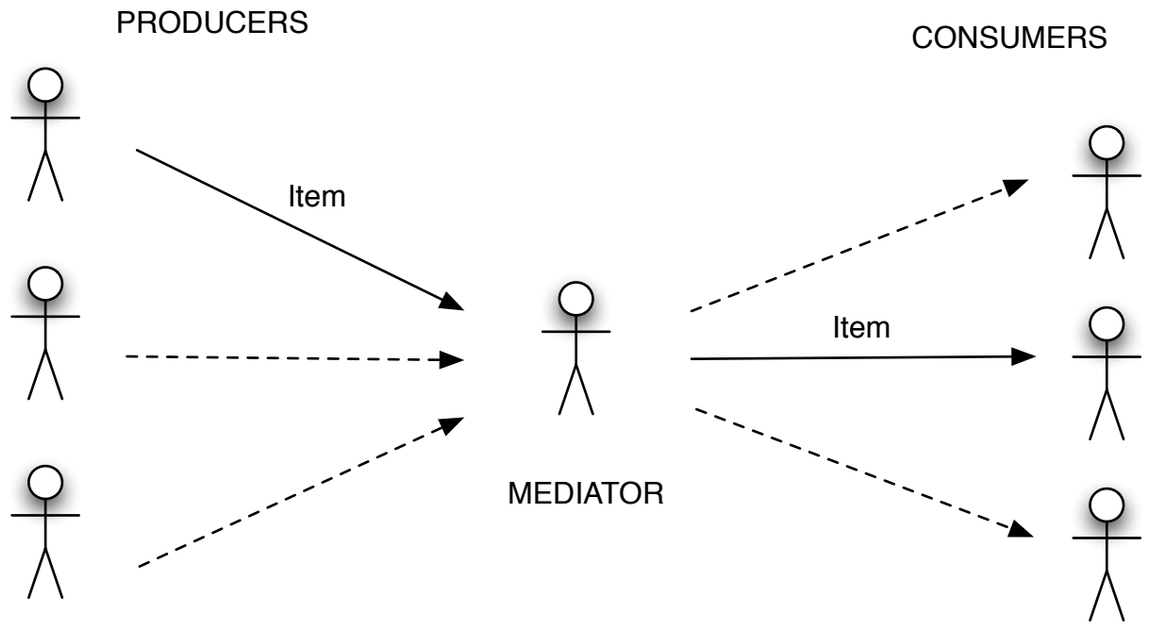
- Use an Object as a shared buffer reduces the number of interactions



- Which kind of interaction? From Agents to (Java) Objects? RPC?
- Not the same level of abstraction for domain entities of the system

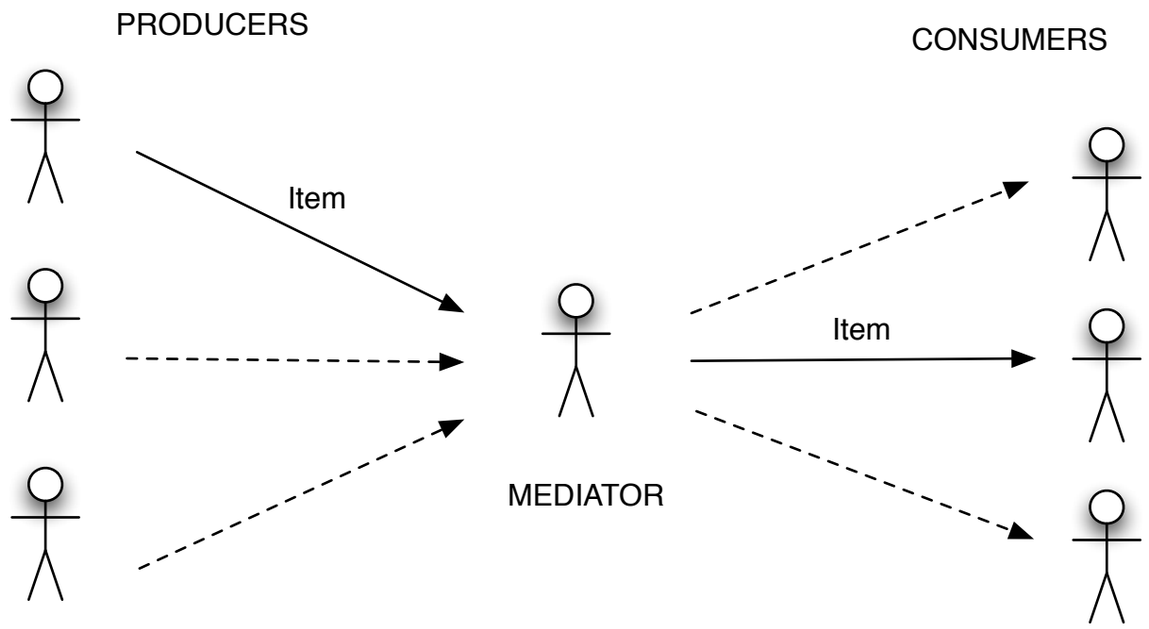
PRODUCERS CONSUMERS IN MAS

- Use a Mediator Agent to maintain the abstraction



PRODUCERS CONSUMERS IN MAS

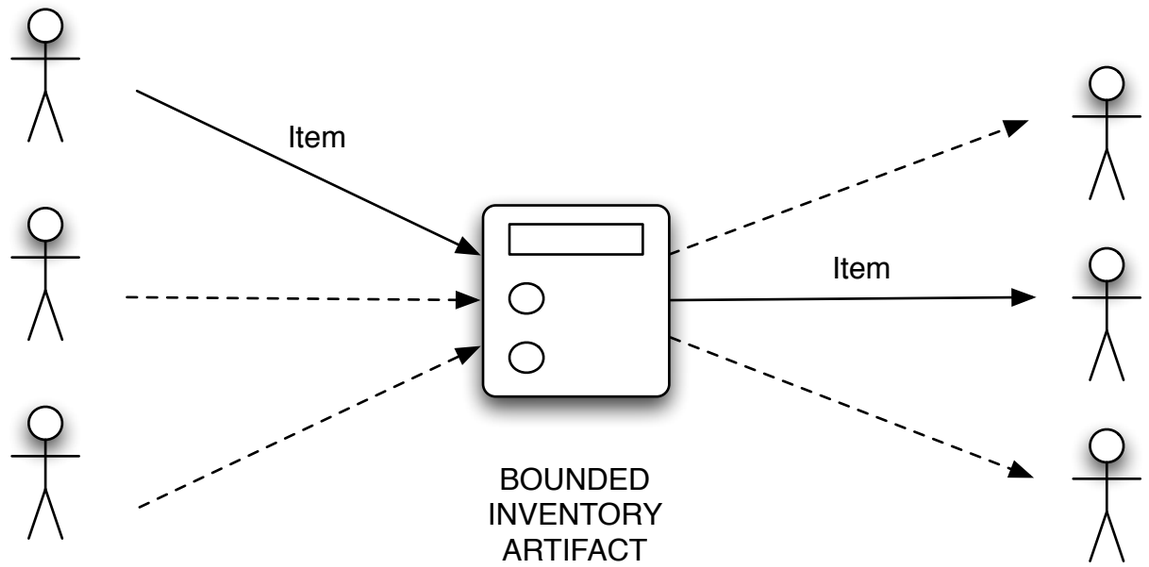
- Use a Mediator Agent to maintain the abstraction



- But which kind of Agents? Reactive? Autonomous? (*can say no*?!?)
- Still need protocols and messages to exchange: opennes?

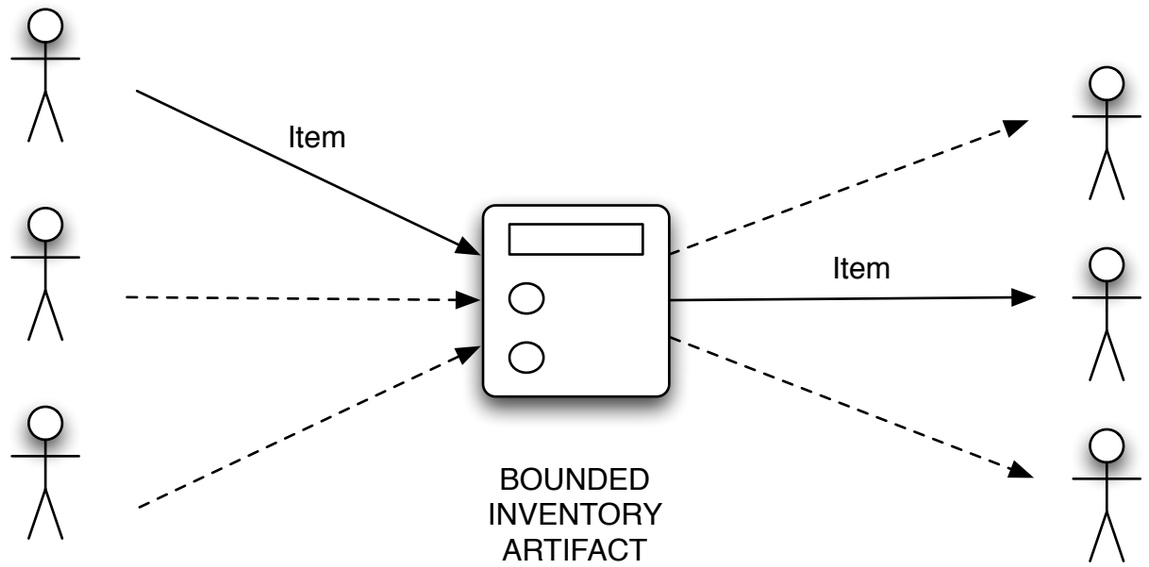
PRODUCERS CONSUMERS IN MAS

- Use an Artifact !



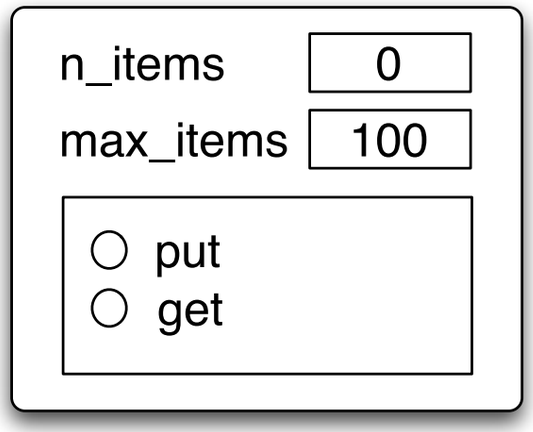
PRODUCERS CONSUMERS IN MAS

- Use an Artifact !



- A Bounded Inventory Artifact is designed to function as a shared Inventory mediating the activities in an authomatic fashion
  - Ruling and Enabling the exchange
- Inventory can be modified to change performance of the system (i.e. capacity, size, data preprocessing)

BOUNDED INVENTORY



OBSERVABLE PROPERTIES:

n_items: int+
max_items: int

Invariants:
n_items <= max_items

USAGE INTERFACE:

put(*item*:Item) / (n_items < max_items):
[obs_prop_updated, op_exec_completed]

get / (n_items >= 0) :
[obs_prop_updated, *new_item*(item:Item),
op_exec_completed]

```
import alice.cartago.*;
import java.util.*;

public class BoundedInventory extends Artifact {
  private LinkedList<Item> items;

  @OPERATION void init(int nmax){
    items = new LinkedList<Item>();
    defineObsProperty("max_items",nmax);
    defineObsProperty("n_items",0);
  }

  @OPERATION(guard="inventoryNotFull") void put(Item obj){
    items.add(obj);
    updateObsProperty("n_items",items.size()+1);
  }

  @OPERATION(guard="itemAvailable") void get(){
    Item item = items.removeFirst();
    updateObsProperty("n_items",items.size()-1);
    signal("new_item",item);
  }

  @GUARD boolean itemAvailable(){ return items.size() > 0; }

  @GUARD boolean inventoryNotFull(Item obj){
    int maxItems = getObsProperty("max_items").intValue();
    return items.size() < maxItems;
  }
}
```

Operational Function

```
+!produceItems : nextItemToProduce(Item)
   <- cartago.lookupArtifact("my-inv", InvID);
      cartago.use(InvID, put(Item), 500).

/* Fail Event */
-!produceItems: true
   <- cartago.use(console,
      println("Insertion failed due to timeout.")).
```

```
+!consume
     : myInventory(InvID) & mySensor(S)
      <- cartago.use(InvID, get, S, 1000);
         cartago.sense(S, new_item(Item));
         !consumeItem(Item);
         !consume.

+!consumeItem(Item) : true
      <- ...
```

Operational Function

```
+!produceItems : nextItemToProduce(Item)
   <- cartago.lookupArtifact("my-inv", InvID);
      cartago.use(InvID, put(Item), 500).

/* Fail Event */
-!produceItems: true
   <- cartago.use(console,
      println("Insertion failed due to timeout.")).
```

```
+!consume
     : myInventory(InvID) & mySensor(S)
      <- cartago.use(InvID, get, S, 1000);
         cartago.sense(S, new_item(Item));
         !consumeItem(Item);
         !consume.

+!consumeItem(Item) : true
      <- ...
```

- Both Producer and Consumer Goals are in Artifact outcomes, after the use of its operations

Operational Function

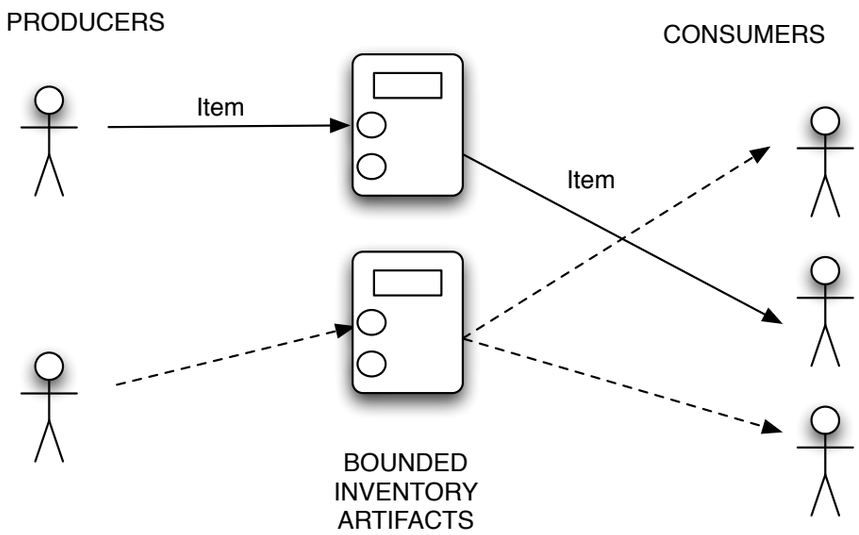
```
+!produceItems : nextItemToProduce(Item)
   <- cartago.lookupArtifact("my-inv", InvID);
      cartago.use(InvID, put(Item), 500).

/* Fail Event */
-!produceItems: true
   <- cartago.use(console,
      println("Insertion failed due to timeout.")).
```

```
+!consume
    : myInventory(InvID) & mySensor(S)
     <- cartago.use(InvID, get, S, 1000);
        cartago.sense(S, new_item(Item));
        !consumeItem(Item);
        !consume.

+!consumeItem(Item) : true
     <- ...
```

- Both Producer and Consumer Goals are in Artifact outcomes, after the use of its operations
- All the computation required to synchronise interactions is externalised in the Bounded Inventory
  - It can possibly pre-process the items, e.g. translate them to different ontologies

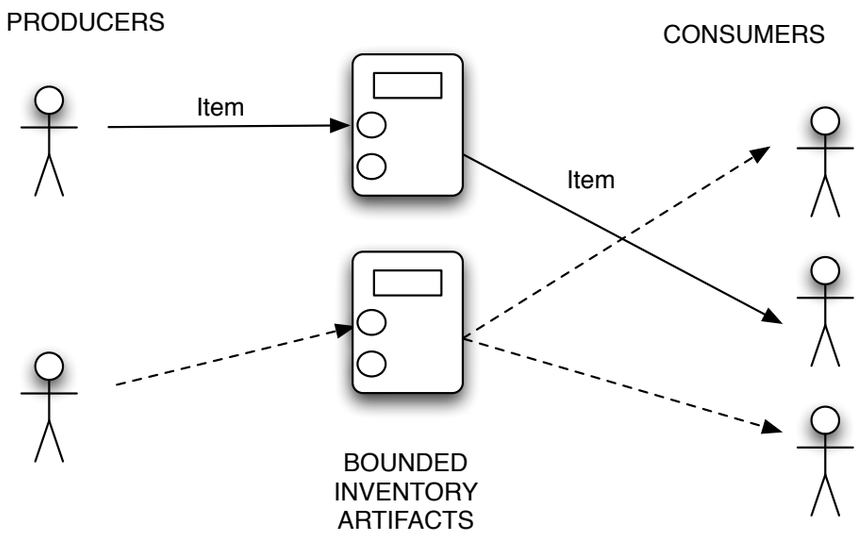
Doxastic Function

- Immagine now that 2 bounded buffers are in place, each with a different ‘load’



Doxastic Function

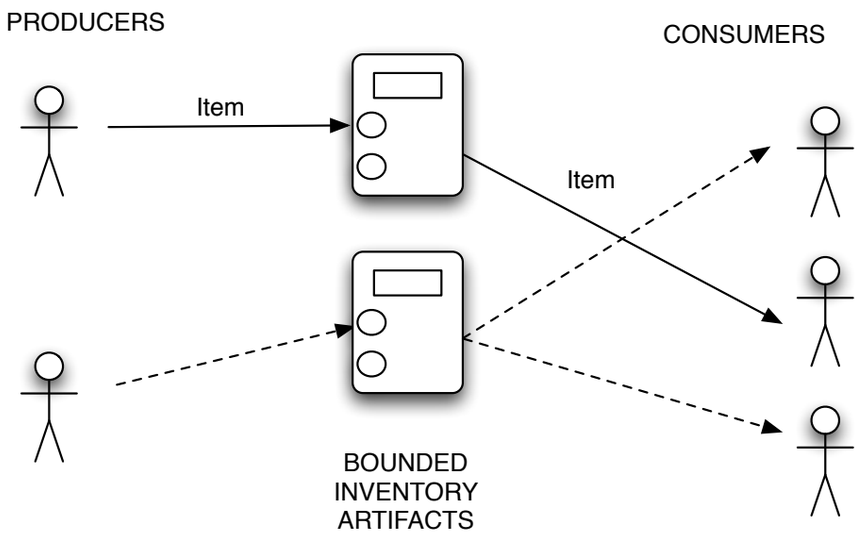
- Immagine now that 2 bounded buffers are in place, each with a different ‘load’




```
+!consumeActivity : true
    <- +min_items(-1);
       cartago.lookupArtifact("inventory-1", InvID1);
       cartago.focus(InvID1);
       cartago.lookupArtifact("inventory-2", InvID2);
       cartago.focus(InvID2);
       +selectedInv(InvID1,0);
       !consumeAction.

/* Updates Goal Supporting Beliefs */
+n_items(N) [source(percept), artifact(InventoryID)]
  : selectedInv(_,N1) & N > N1
    <- -+selectedInv(InventoryID,N).

+!consumeAction : selectedInv(InvID,_)
    <- cartago.use(InvID, get, mySensor);
       cartago.sense(mySensor, new_item(Item));
       cartago.use(console,println("Consumed:",Item));
       !consumeAction.
```

Doxastic Function

- Immagine now that 2 bounded buffers are in place, each with a different ‘load’

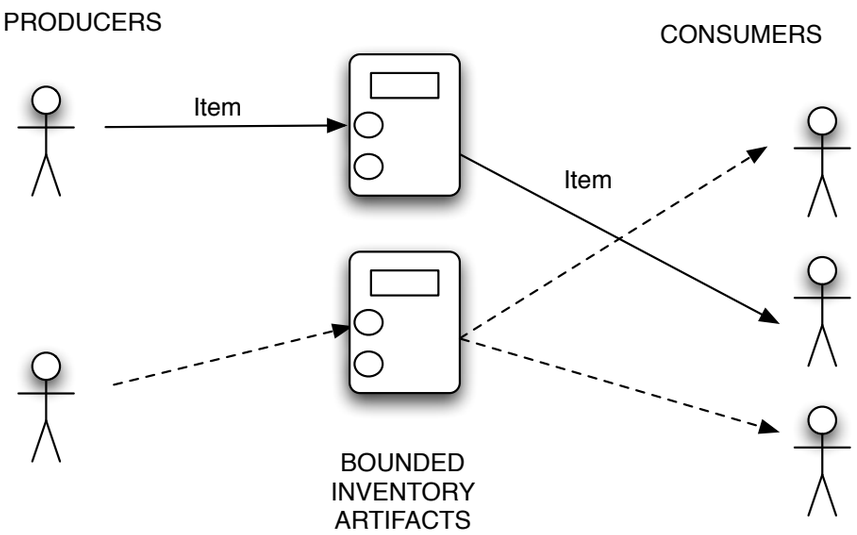



```
+!consumeActivity : true
    <- +min_items(-1);
       cartago.lookupArtifact("inventory-1", InvID1);
       cartago.focus(InvID1);
       cartago.lookupArtifact("inventory-2", InvID2);
       cartago.focus(InvID2);
       +selectedInv(InvID1,0);
       !consumeAction.

/* Updates Goal Supporting Beliefs */
+n_items(N) [source(percept), artifact(InventoryID)]
  : selectedInv(_,N1) & N > N1
    <- -+selectedInv(InventoryID,N).

+!consumeAction : selectedInv(InvID,_)
    <- cartago.use(InvID, get, mySensor);
       cartago.sense(mySensor, new_item(Item));
       cartago.use(console,println("Consumed:",Item));
       !consumeAction.
```

- By focusing on artifact properties, Consumer can decide which is the most reliable one to use (Goal Supporting Beliefs)

Doxastic Function

- Immagine now that 2 bounded buffers are in place, each with a different ‘load’




```
+!consumeActivity : true
    <- +min_items(-1);
       cartago.lookupArtifact("inventory-1", InvID1);
       cartago.focus(InvID1);
       cartago.lookupArtifact("inventory-2", InvID2);
       cartago.focus(InvID2);
       +selectedInv(InvID1,0);
       !consumeAction.

/* Updates Goal Supporting Beliefs */
+n_items(N) [source(percept), artifact(InventoryID)]
  : selectedInv(_,N1) & N > N1
    <- -+selectedInv(InventoryID,N).

+!consumeAction : selectedInv(InvID,_)
    <- cartago.use(InvID, get, mySensor);
       cartago.sense(mySensor, new_item(Item));
       cartago.use(console,println("Consumed:",Item));
       !consumeAction.
```

- By focusing on artifact properties, Consumer can decide which is the most reliable one to use (Goal Supporting Beliefs)
- Update of Goal Supporting Beliefs is done automatically
  - in reaction to a property change event

Cognitive Artifacts

The notion is related to agents able to bring about artifact informational functions [Norman]

Cognitive Artifacts

The notion is related to agents able to bring about artifact informational functions [Norman]

- 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 artifact informational functions [Norman]

- 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

Two more Steps towards an *Extended Mind [Clarks]*

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

- Artifact discovery as a process of **Epistemic Reasoning**
  - Agents need to *Read, learn,* and *match* Artifact Representational contents

- 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)

ONGOING EVALUATION (APPLICATIONS)

- ORA4MAS (w/ Hubner, Boissier, Kitio, Ricci)
  - exploiting artifacts to build an organisational infrastructure

- CArtAgO-WS
  - basic set of artifacts for building SOA/WS applications
    - interacting with web services
    - implementing web services

- FORMAL MODEL
  - Interactions
- ARTIFACT LIBRARIES
  - setting up a set of reusable artifacts in MAS applications

REFERENCES

Omicini, A., Ricci, A., Viroli, M.: Agens Faber: Toward A Theory of Artefacts for MAS *Electronic Notes in Theoretical Computer Sciences, Elsevier Science B.V.,* **2006***, 150*, 21-36

Ricci, A., Viroli, M., Omicini, A.: 'Give Agents their Artifacts': The A&A Approach for Engineering Working Environments*. In 6th international Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007). Honolulu, Hawai'i, USA,* **2007**

Ricci, A., Viroli, M. , Omicini, A. The A&A Programming Model & Technology for Developing Agent Environments in MAS. *ProMAS'07,Post-proceedings, Springer,* **2007***, 4908*, 91-109

Piunti, M., Ricci, A. Cognitive Artifacts for Intelligent Agents in MAS: Exploiting Relevant Information residing in Environments*. Workshop on Knowledge Representation for Agents and Multi-Agent Systems (KRAMAS 2008), Sydney,* **2008**

Ricci, A., Piunti, M., Acay, L. D., Bordini, R.; Hubner, J. , Dastani, M. Integrating Artifact-Based Environments with Heterogeneous Agent-Programming Platforms*. In 6th international Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'08). Estoril,* **2008**

Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems*. Journal of Autonomous Agents and Multi-Agent Systems,* **2008***, 17(3)*

Piunti, M.; Ricci, A.; Braubach, L. & Pokahr, A. Goal-Directed Interactions in Artifact-Based MAS: Jadex Agents playing in CArtAgO Environments *In IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.* Sydney. **2008**

Piunti, M., Ricci, A. From Agents to Artifacts Back and Forth: Operational and Doxastic Use of Artifacts in MAS*. In Proceedings of Sixth European Workshop on Multi-Agent Systems, to appear.* **2008**