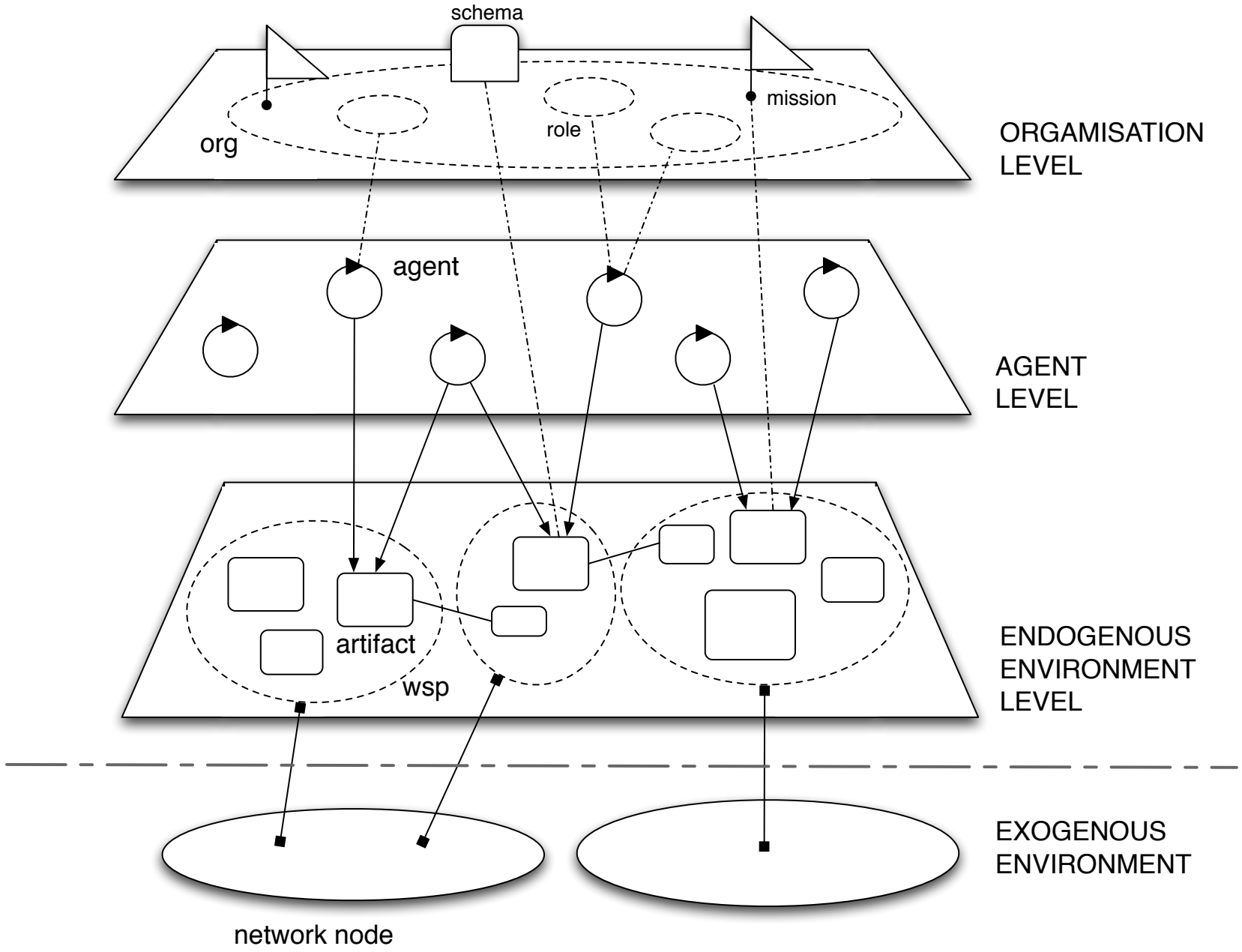
Environment Oriented Programming with CArtAgO

Jomi F. Hübner

Federal University of Santa Catarina, Brazil

PPGEAS 2017 — UFSC

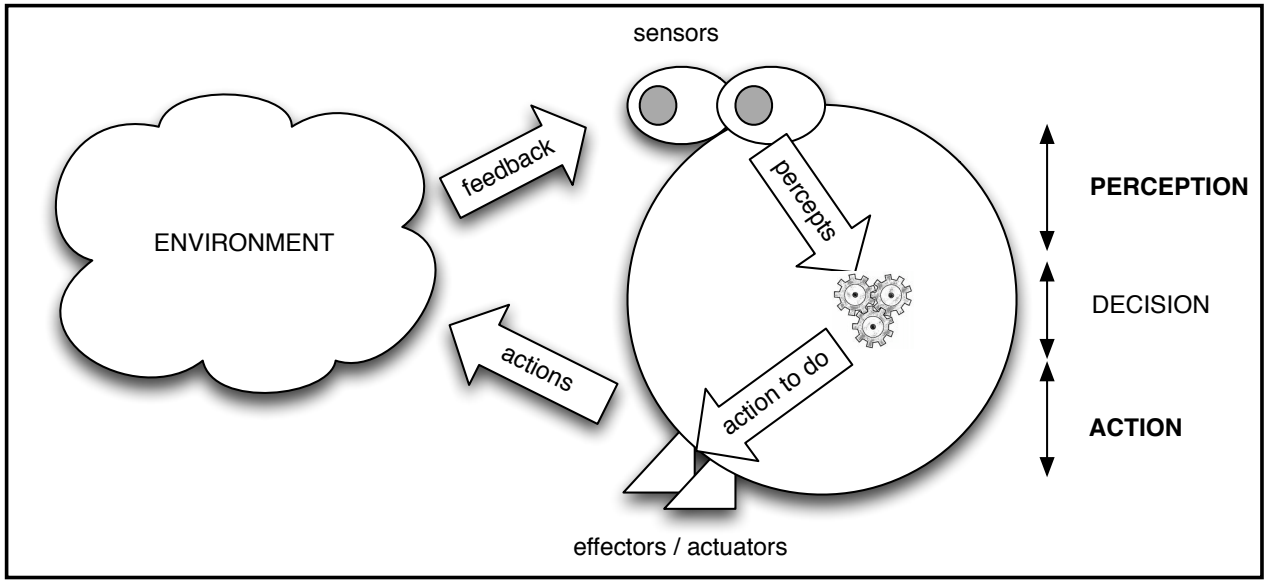
Multi-Agent System (our perspective)



Back to the Notion of Environment in MAS

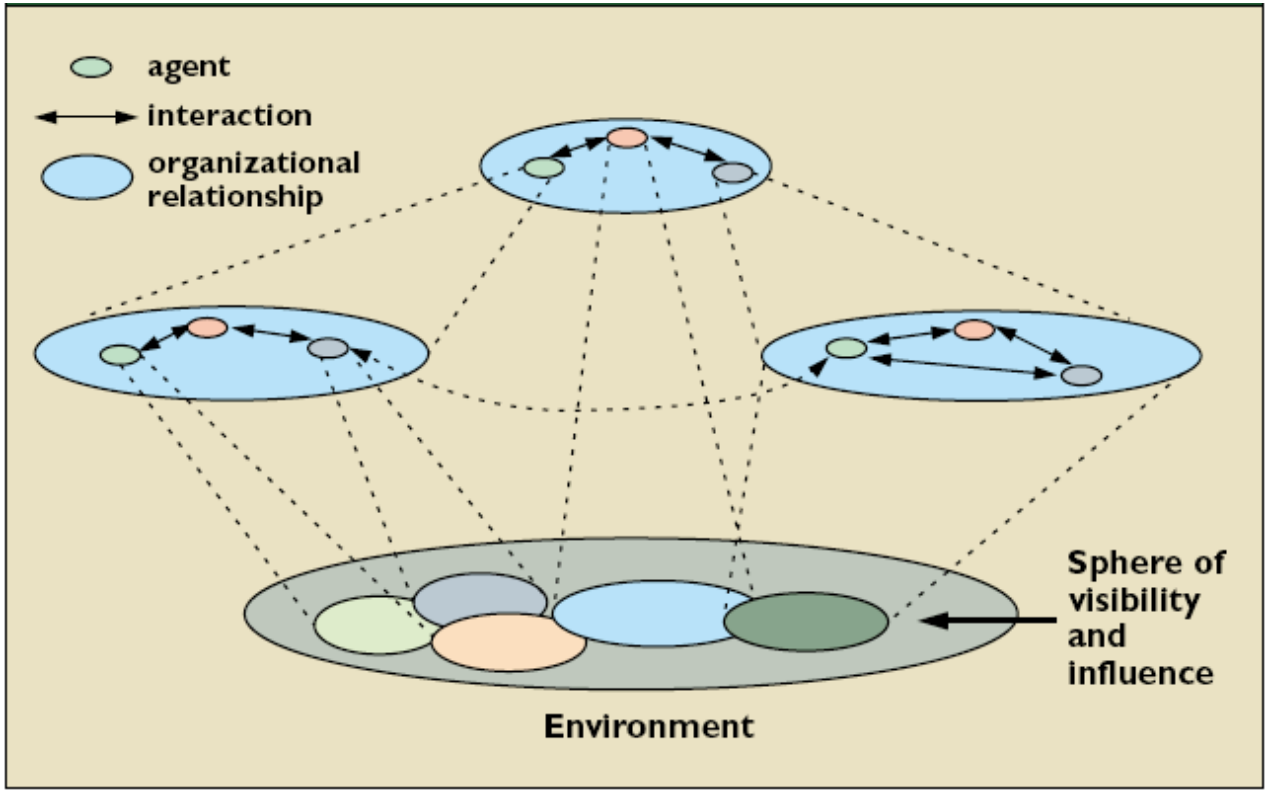
- The notion of environment is intrinsically related to the notion of agent and multi-agent system
  - "An agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objective" [Wooldridge, 2002]
  - "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon the environment through effectors. " [Russell and Norvig, 2003]
- Including both physical and software environments

Single Agent Perspective



- Perception
  - process inside agent inside of attaining awareness or understanding sensory information, creating percepts perceived form of external stimuli or their absence
- Actions
  - the means to affect, change or inspect the environment

Multi-Agent Perspective



- In evidence
  - overlapping spheres of visibility and influence
  - ..which means: interaction

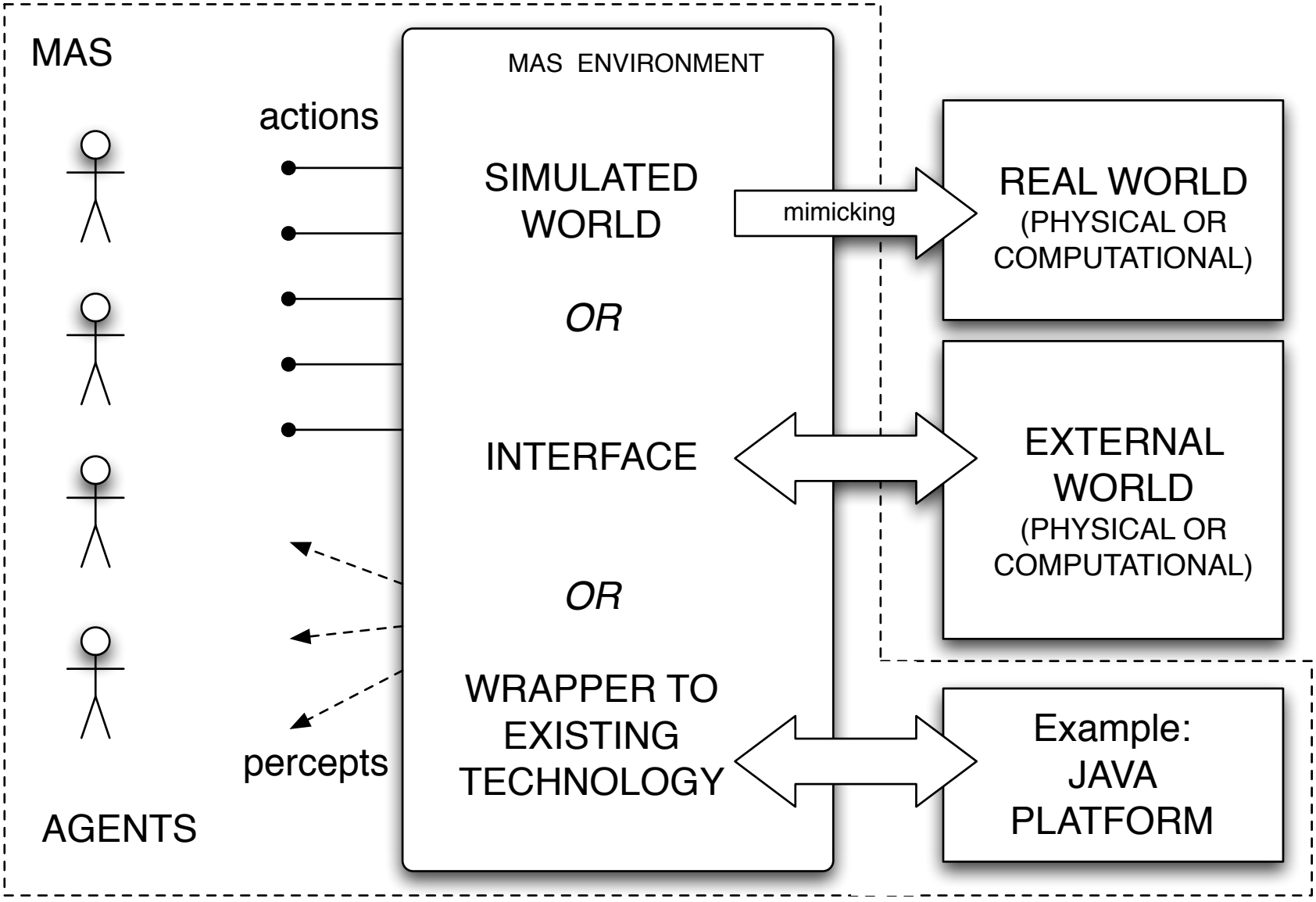
Why Environment Programming

- Basic level
  - to create testbeds for real/external environments
  - to ease the interface/interaction with existing software environments
- Advanced level
  - to uniformly encapsulate and modularise functionalities of the MAS out of the agents
    - typically related to interaction, coordination, organisation, security
    - externalisation
  - this implies changing the perspective on the environment
    - environment as a first-class abstraction of the MAS
    - endogenous environments (vs. exogenous ones)
    - programmable environments

Environment Programming: General Issues

- Defining the interface
  - actions, perceptions
  - data-model
- Defining the environment computational model & architecture
  - how the environment works
  - structure, behaviour, topology
  - core aspects to face: concurrency, distribution
- Defining the environment programming model
  - how to program the environment

Basic Level Overview

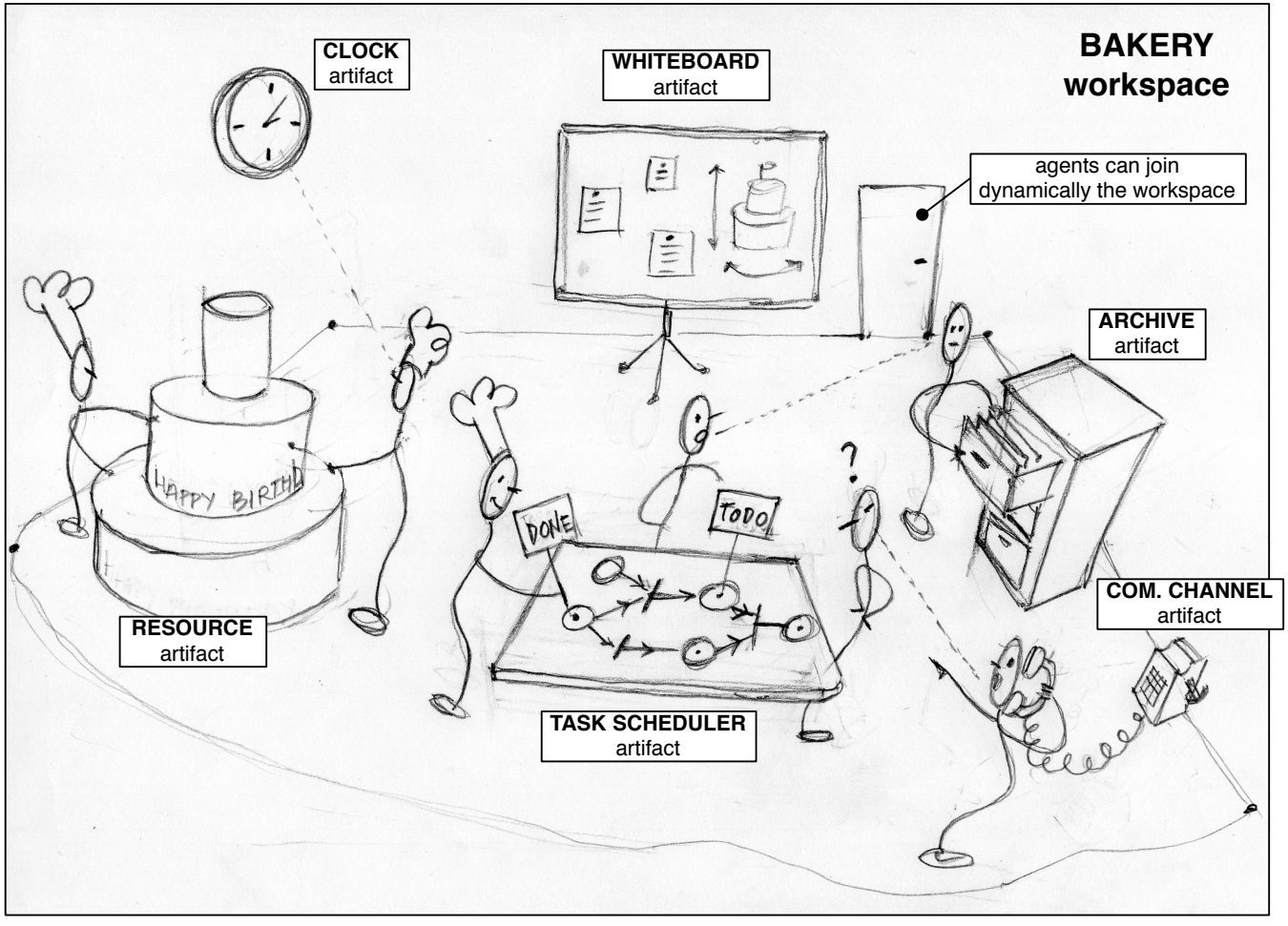


Advanced Level Overview

- Vision: environment as a first-class abstraction in MAS [Weyns et al., 2007, Ricci et al., 2010b]
  - application or endogenous environments, i.e. that environment which is an explicit part of the MAS
  - providing an exploitable design & programming abstraction to build MAS applications
- Outcome
  - distinguishing clearly between the responsibilities of agent and environment
    - separation of concerns
  - improving the engineering practice

A&A and CArtAgO

Agents and Artifacts (A&A) Conceptual Model: Background Human Metaphor



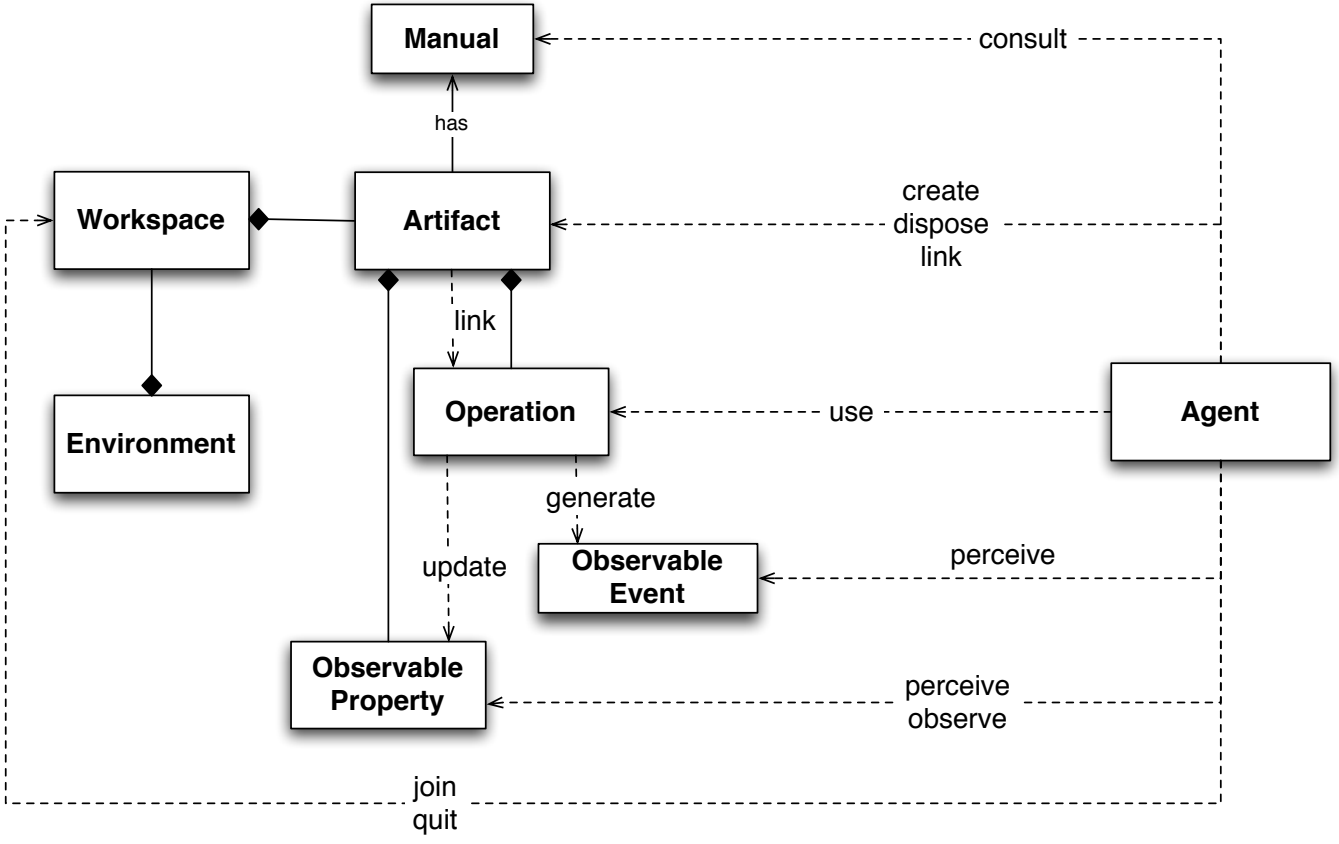
A&A Basic Concepts [Omicini et al., 2008]

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

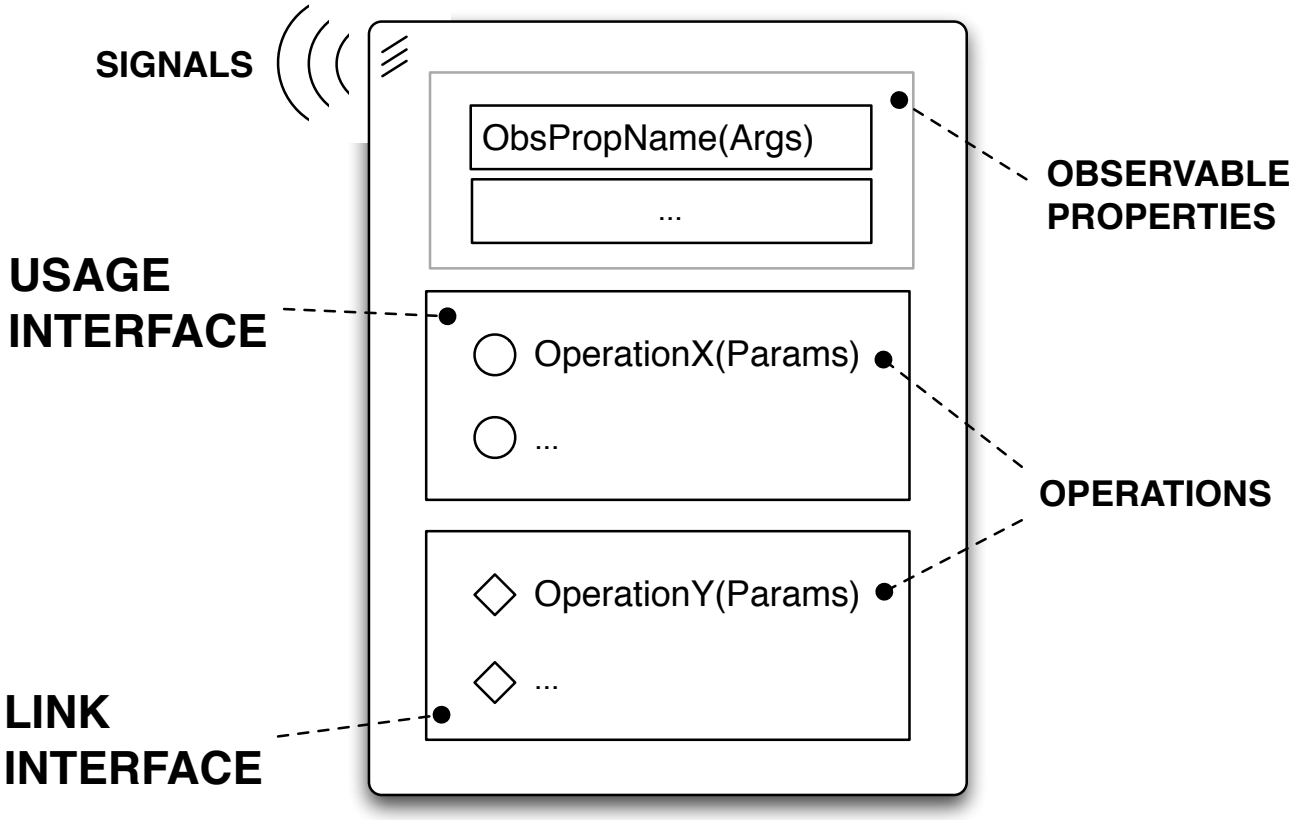
A&A Programming Model Features [Ricci et al., 2007b]

- Abstraction
  - artifacts as first-class resources and tools for agents
- Modularisation
  - artifacts as modules encapsulating functionalities, organized in workspaces
- Extensibility and openness
  - artifacts can be created and destroyed at runtime by agents
- Reusability
  - artifacts (types) as reusable entities, for setting up different kinds of environments

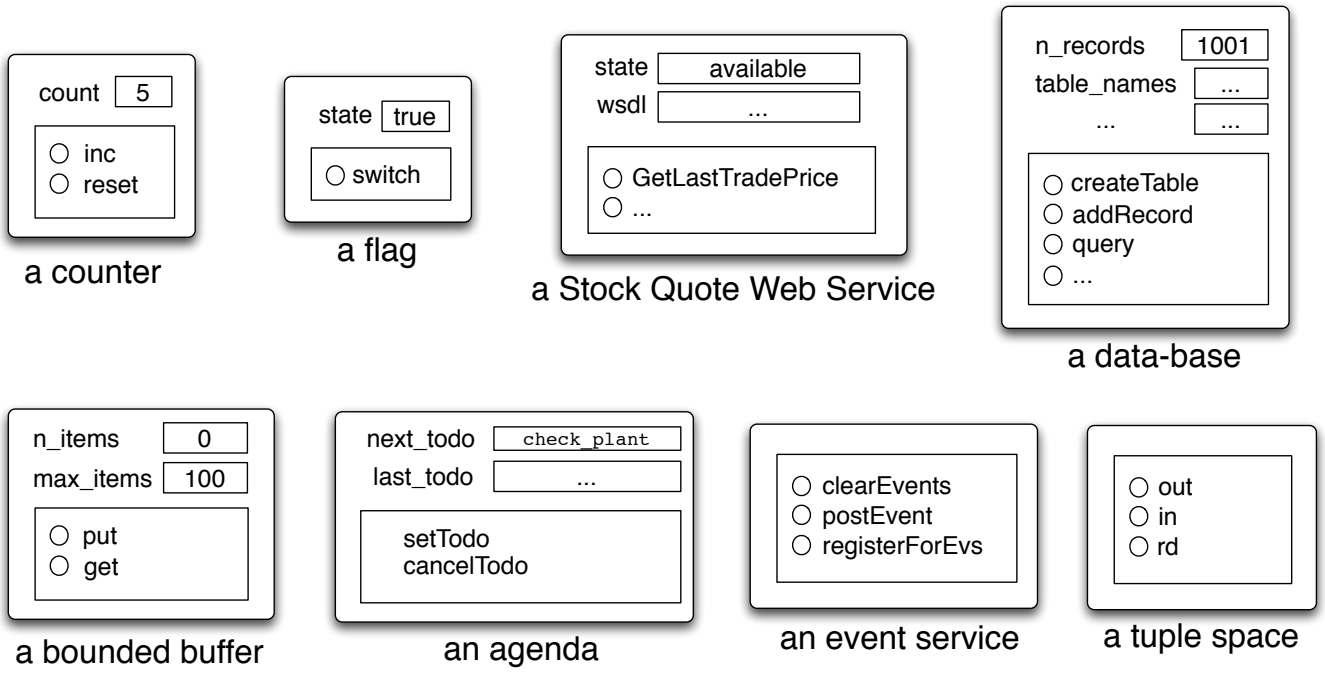
A&A Meta-Model in More Detail [Ricci et al., 2010b]



Artifact Abstract Representation



A World of Artifacts



A Simple Taxonomy

- Individual or personal artifacts
  - designed to provide functionalities for a single agent use
    - e.g. an agenda for managing deadlines, a library...
- Social artifacts
  - designed to provide functionalities for structuring and managing the interaction in a MAS
  - coordination artifacts [Omicini et al., 2004], organisation artifacts, ...
    - e.g. a blackboard, a game-board,...
- Boundary artifacts
  - to represent external resources/services
    - e.g. a printer, a Web Service
  - to represent devices enabling I/O with users
    - e.g GUI, console, etc.

Actions and Percepts in Artifact-Based Environments

- Explicit semantics defined by the (endogenous) environment [Ricci et al., 2010c]
  - success/failure semantics, execution semantics
  - defining the contract (in the SE acceptation) provided by the environment

actions $\longleftrightarrow$ artifacts' operation

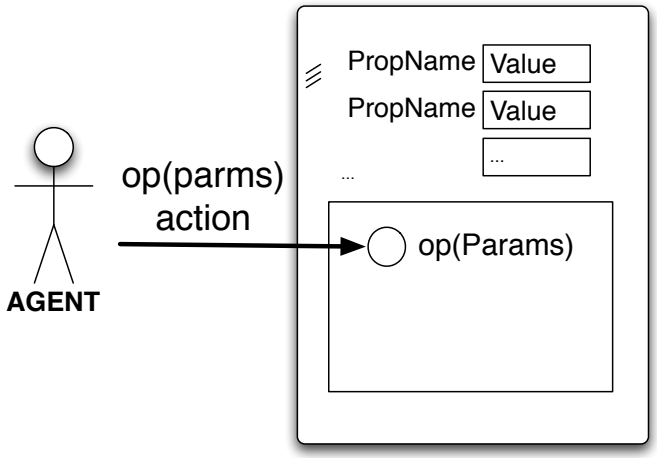
the action repertoire is given by the dynamic set of operations provided by the overall set of artifacts available in the workspace can be changed by creating/disposing artifacts

- action success/failure semantics is defined by operation semantics

percepts $\longleftrightarrow$ artifacts' observable properties + signals

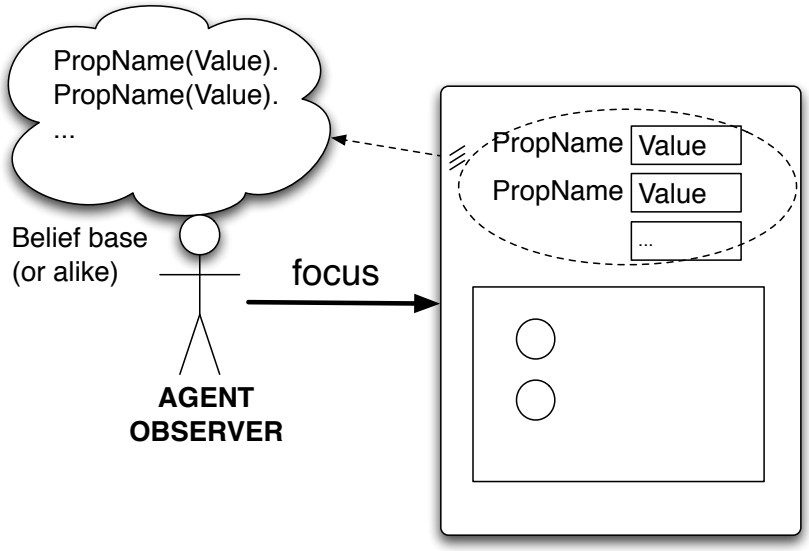
properties represent percepts about the state of the environment signals represent percepts concerning events signalled by the environment

Interaction Model: Use



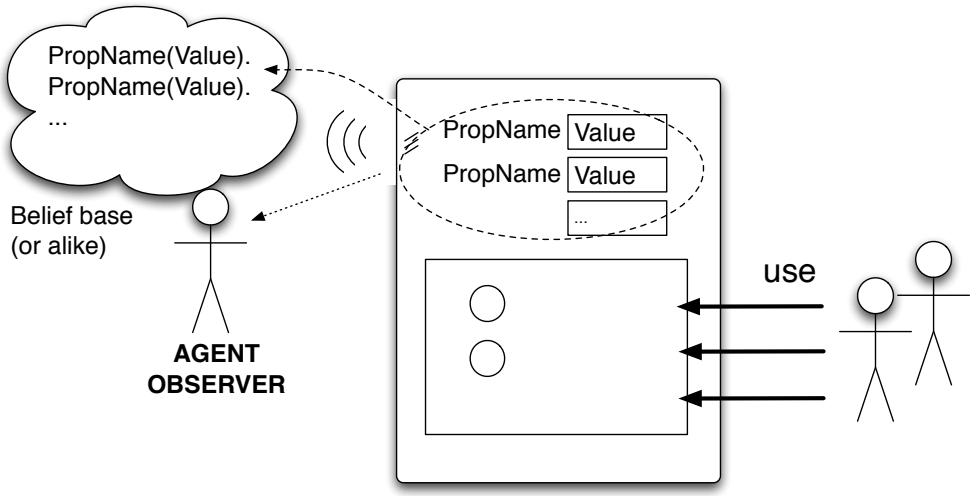
- Performing an action corresponds to triggering the execution of an operation
  - acting on artifact's usage interface

Interaction Model: Observation



- Agents can dynamically select which artifacts to observe
  - predefined `focus`/`stopFocus` actions

Interaction Model: Observation



- By focussing an artifact
  - observable properties are mapped into agent dynamic knowledge about the state of the world, as percepts
    - e.g. belief base
  - signals are mapped as percepts related to observable events

CArtAgO

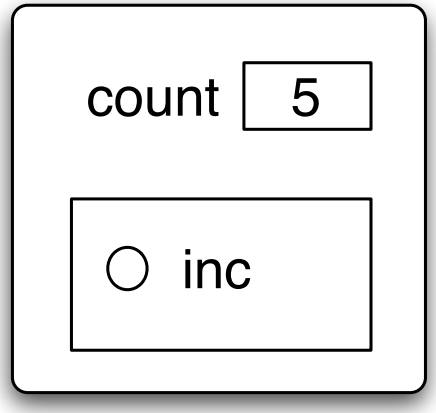
- Common ARtifact infrastructure for AGent Open environment (CArtAgO) [Ricci et al., 2009]
- Computational framework / infrastructure to implement and run artifact-based environment [Ricci et al., 2007c]
  - Java-based programming model for defining artifacts
  - set of basic API for agent platforms to work within artifact-based environment
- 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 `https://github.com/CArtAgO-lang/cartago`

Example 1: A Simple Counter Artifact

```
class Counter extends Artifact {

  void init(){
    defineObsProp("count",0);
  }

  @OPERATION void inc(){
    ObsProperty p = getObsProperty("count");
    p.updateValue(p.intValue() + 1);
    signal("tick");
  }
}
```



- Some API spots
  - `Artifact` base class
  - `@OPERATION` annotation to mark artifact's operations
  - set of primitives to work define/update/.. observable properties
  - `signal` primitive to generate signals

Example 1: User and Observer Agents

USER(S)

```
!create_and_use.

+!create_and_use : true
  <- !setupTool(Id);
     // use
     inc;
     // second use specifying the Id
     inc [artifact_id(Id)].

// create the tool
+!setupTool(C): true
  <- makeArtifact("c0","Counter",C).
```

OBSERVER(S)

```
!observe.

+!observe : true
  <- ?myTool(C);  // discover the tool
     focus(C).

+count(V)
  <- println("observed new value: ",V).

+tick [artifact_name(Id,"c0")]
  <- println("perceived a tick").

+?myTool(CounterId): true
  <- lookupArtifact("c0",CounterId).

-?myTool(CounterId): true
  <- .wait(10);
     ?myTool(CounterId).
```

- Working with the shared counter

Action Execution & Blocking Behaviour

- Given the action/operation map, by executing an action the intention/activity is suspended until the corresponding operation has completed or failed
  - action completion events generated by the environment and automatically processed by the agent/environment platform bridge
  - no need of explicit observation and reasoning by agents to know if an action succeeded
- However the agent execution cycle is not blocked!
  - the agent can continue to process percepts and possibly execute actions of other intentions

Summary

- environment as a first-class abstraction of the MAS
  - endogenous environments (vs. exogenous ones)
  - programmable environments
- encapsulate functionalities of the MAS out of the agents
  - externalisation