Multi-agents Coordination

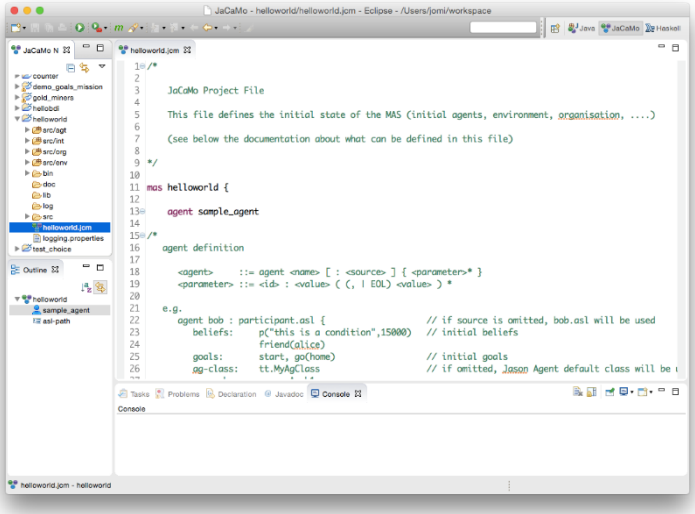
Environment-based Coordination

Outline

- **Environment-based Coordination**
  - A concept with several meanings
  - Environment and Coordination
  - Shared spaces systems example and application

Environment
A concept with several meanings

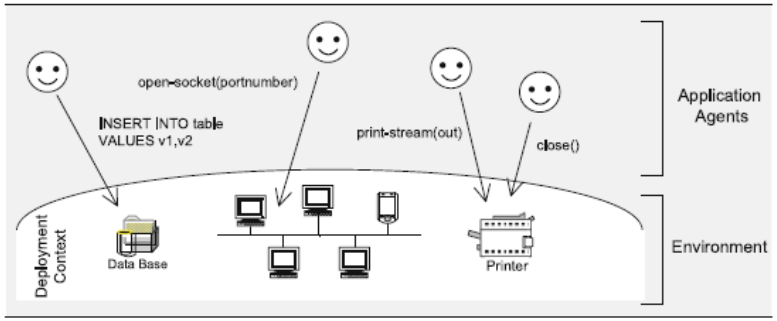
- **Environment for MAS design and processing**
  - Run-time support
  - MAS development tool



- **Computational resource environment**
  - Software infrastructure on which the MAS is executed
  - Hardware infrastructure on which the MAS runs

Environment
A concept with several meanings

- Entities and resources outside the agent system



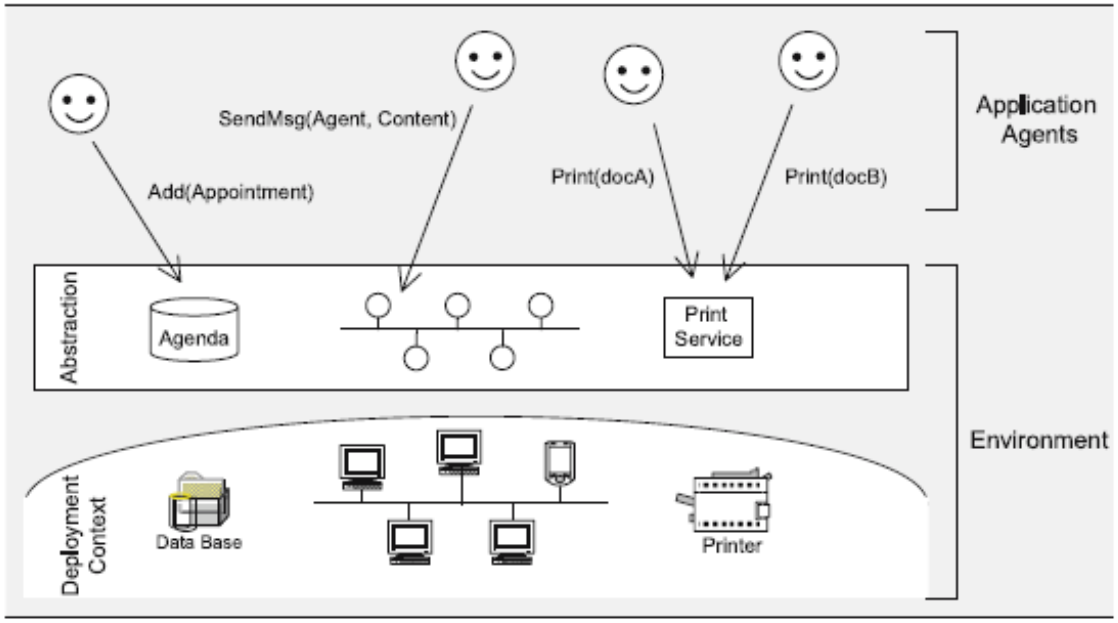
Weyns, D., Omicini, A., & Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous agents and multi-agent systems*, *14*(1), 5-30.

- Logical entity in which agents and other objects/resources are embedded

MINES Saint-Étienne

Environment
A concept with several meanings

- **“The environment can provide agents with an abstraction level that shields low-level details of the deployment context—as well as other resources in the system.”**

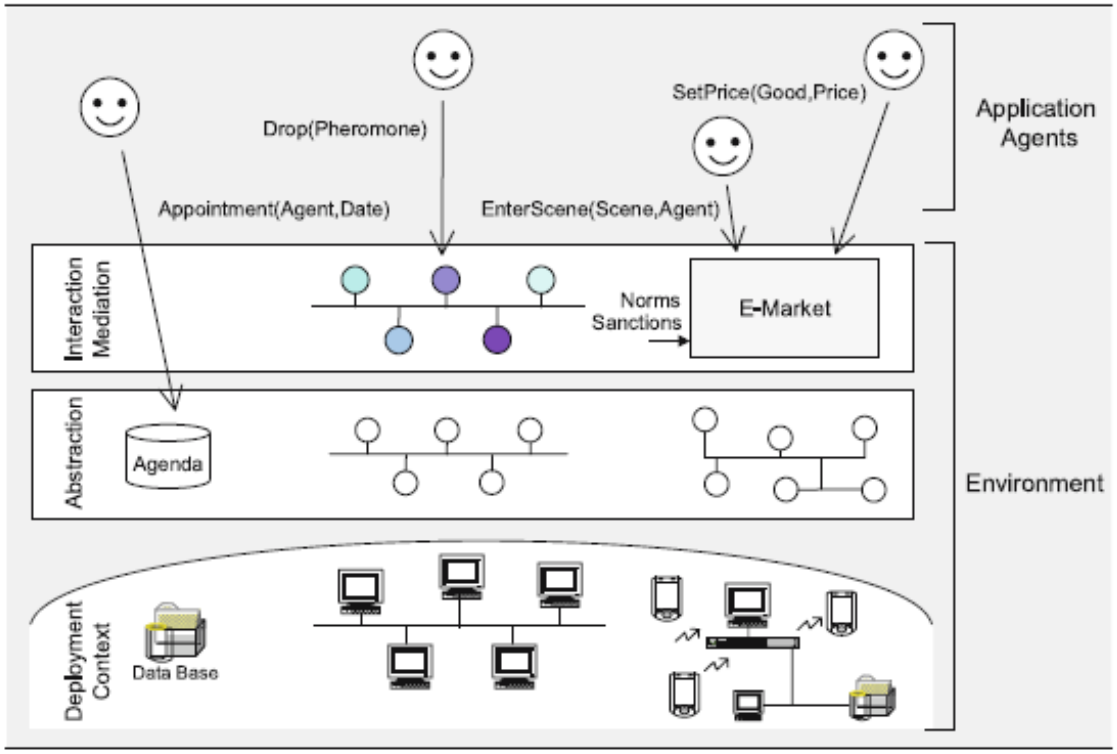


JACAMO Artefact

Weyns, D., Omicini, A., & Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous agents and multi-agent systems*, *14*(1), 5-30.

Environment
A concept with several acceptations

- **“Environment can provide an interaction-mediation level to support mediated interaction in the environment.”**



Weyns, D., Omicini, A., & Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous agents and multi-agent systems*, *14*(1), 5-30.

Jacamo

```java
package tools;

import cartago.Artifact;

/**
 *      Artifact that implements the auction.
 */
public class AuctionArt extends Artifact {


    @OPERATION public void init(String taskDs, int maxValue)  {
        // observable properties
        defineObsProperty("task",          taskDs);
        defineObsProperty("maxValue",      maxValue);
        defineObsProperty("currentBid",    maxValue);
        defineObsProperty("currentWinner", "no_winner");
        System.out.println("Auction Artifact for "+taskDs+" max "+maxValue+" has been initialized");
    }

    @OPERATION public void bid(double bidValue) {
        ObsProperty opCurrentValue  = getObsProperty("currentBid");
        ObsProperty opCurrentWinner = getObsProperty("currentWinner");
        if (bidValue < opCurrentValue.doubleValue()) {  // the bid is better than the previous
            opCurrentValue.updateValue(bidValue);
            opCurrentWinner.updateValue(getCurrentOpAgentId().getAgentName());
        }
    }

}
```

Artefact to encapsulate environment components : library of resources and interaction tools

Read property

Call method

```
i_am_winning(Art)   // check if I placed the current best bid on auction artifact Art
   :- currentWinner(W)[artifact_id(Art)] &
      .my_name(Me) & .term2string(Me,MeS) & W == MeS.


+currentBid(V)[artifact_id(Art)]         // there is a new value for current bid
    : not i_am_winning(Art)  &           // I am not the current winner
      my_price(P) & P < V                // I can offer a better bid
   <- //.print("my bid in auction artifact ", Art, " is ",P);
bid( P ).                               // place my bid offering a cheaper service
```

Environment

A first class abstraction

The environment is a first-class abstraction that provides the surrounding conditions for agents to exist and that mediates both the interaction among agents and the access to resources.

- **The environment must**
  - Provide observability

    The environment provides the representation structures, resources and bodies of agents.
  - Provide accessibility

    The environment manages the accessibility of structures and resources.
  - Support SMA regulation

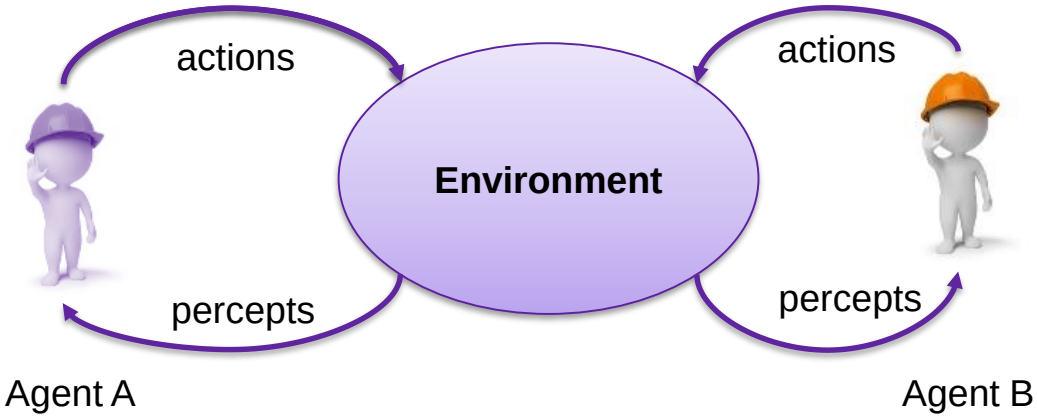
    The environment controls access to structures and resources, it maintains access laws.
  - Provide Ontology

    The environment provides a common ontology that can be consulted by agents.

Weyns, D., Omicini, A., & Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous agents and multi-agent systems*, *14*(1), 5-30.

Environment-based coordination

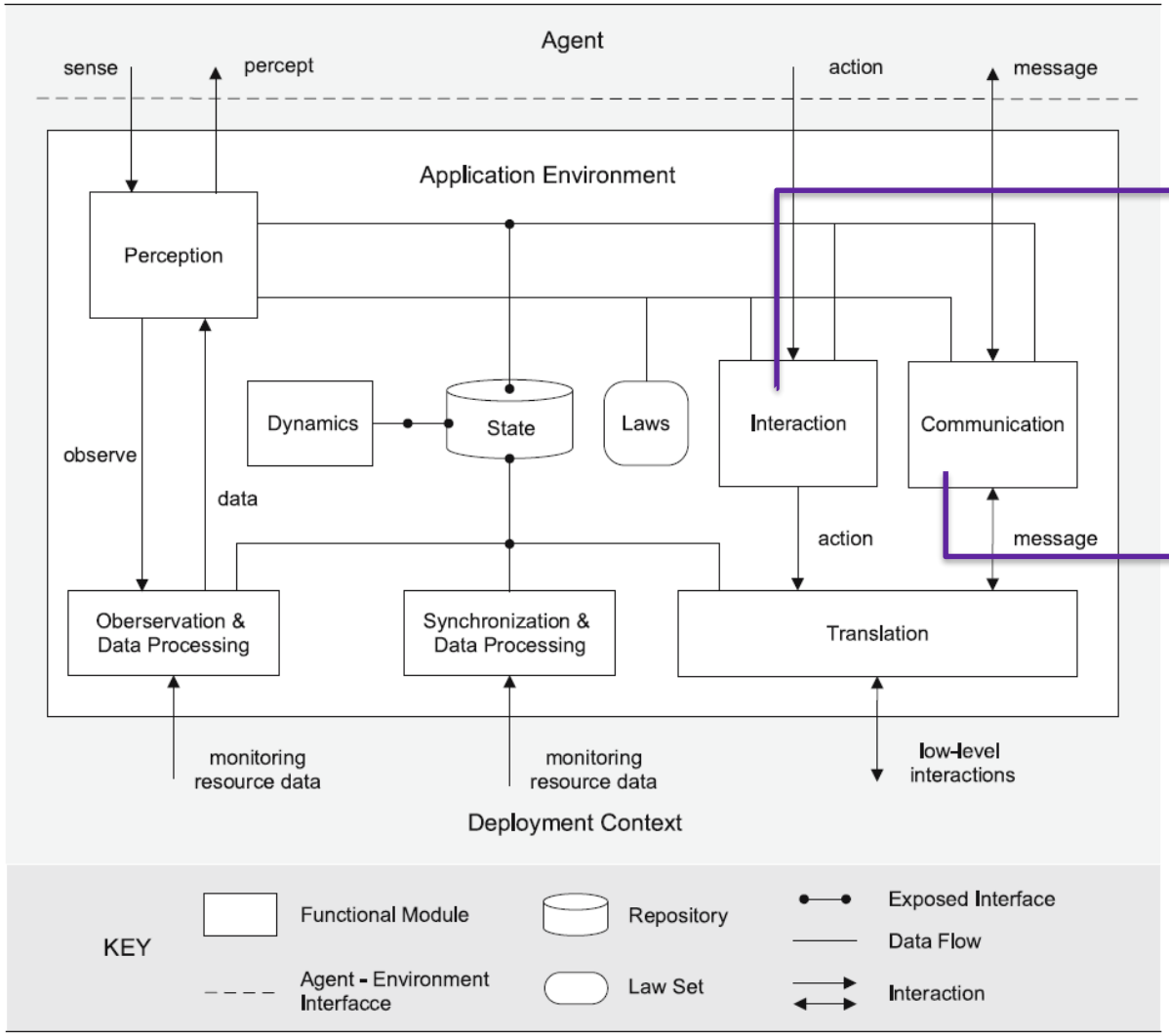
Principle



- **Action / perception coordination mechanism**
  - Agent A actions modify the environment and **indirectly** influence the decision process of B
    - No intention of agent A to influence agent B
    - Agent B decides of its actions according to its context that is defined by the current perception of the environment and his own knowledge.
- **Environment mediates interaction between agents**
  - Agent A could willingly influence the other agents with “coordination actions”
  - Messages are also coordination media that could be mediated by the environment.

Environment

Reference architecture



Shared space (Tuplespace, Blackboard) solutions

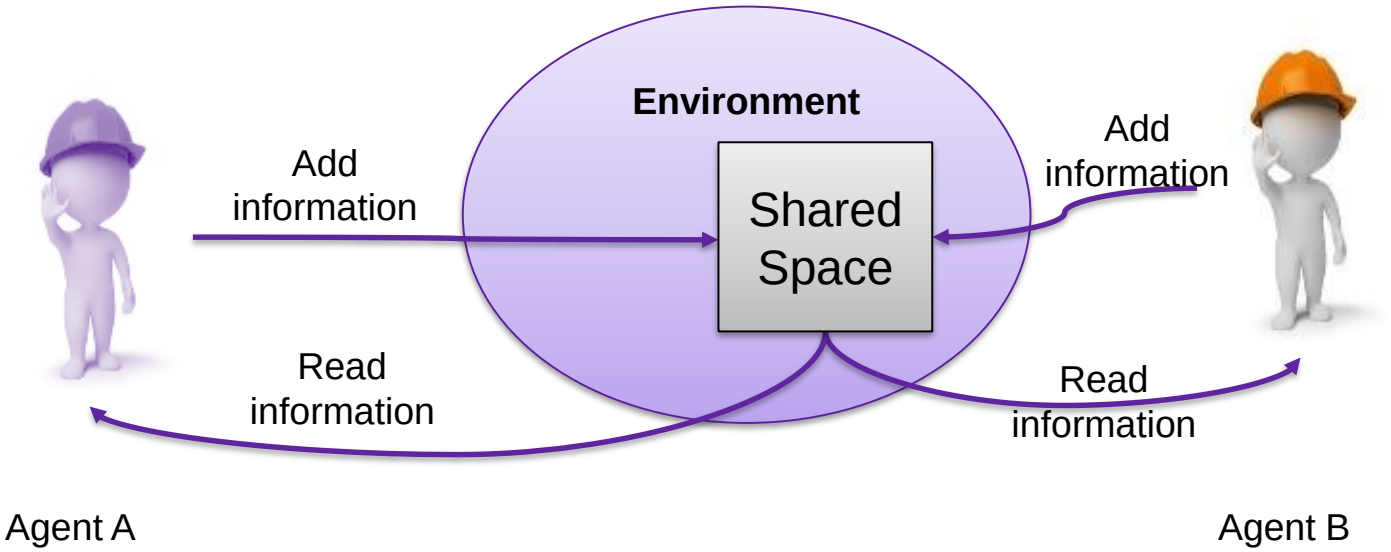
Multi-party communication solutions

Weyns, D., Omicini, A., & Odell, J. (2007). Environment as a first-class abstraction in multiagent systems. *Autonomous agents and multi-agent systems*, *14*(1), 5-30.

Shared Space

Principle

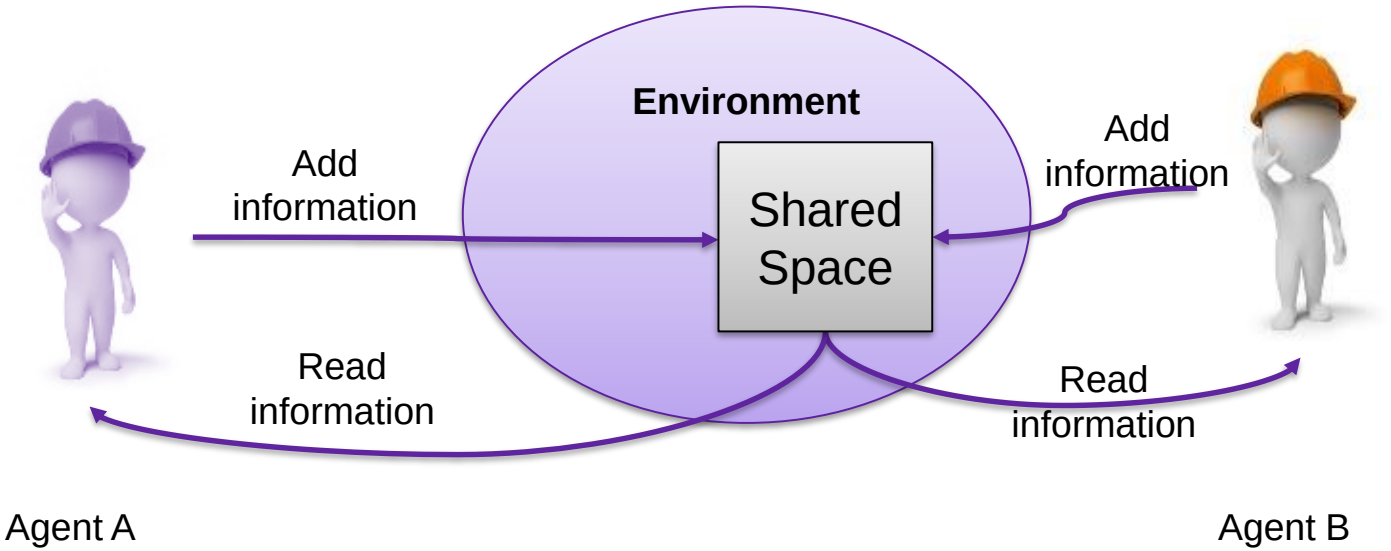
- **A logical structure in the environment for sharing information**



Shared Space

Principle

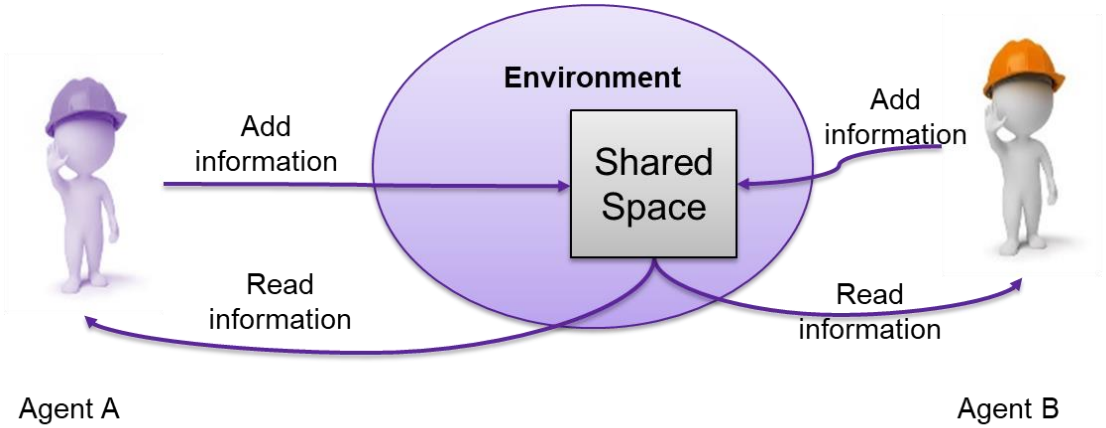
- **Based on indirect interaction, because information are**
  - Put in the logical structure by the agent that wants to share information
  - Read in the logical by the agent(**S**) that are interested by the information



Shared Space

Principle

- **As a result, coordination mechanism based on environment is**
  - Decoupled in space and time
  - Directed by the reader(s): they choose what is relevant for them. The agent putting information does not choose how it will be used.
  - Adapted for open system: a unique (potentially distributed) place where information are shared.
  - Implied a common knowledge about the shared information (syntax, semantic)



Shared Space

Tuple Space

- **Origin: Distributed Systems.**
- **Linda model Principle [Carriero 86]**
  - A shared memory with tuples
  - A matching process for requesting relevant tuples
- **How it works**
  - A tuple is an ordered set of values.
  - A template is a tuple with values and wildcards
  - Matching process returns tuples conformed to the template.
  - Three operators :
    - *out(t)* : add the tuple t,
    - *in(m)* : return a tuple conformed to the template m. The tuple is deleted from the shared space
    - *read(m)* : return a tuple conformed to the template m without its deletion
  - Operators *in* and *read* are blocking operators.
  - Tuplespace communication is generative, i.e., a tuple is independent of the generating process after generation, and can have a lifetime beyond the generating process.

Tuple-Space

Example

- *"Mobile agent paradigm for processing data to merge the IoT, sensor networks, and Cloud-based environments seamless"*

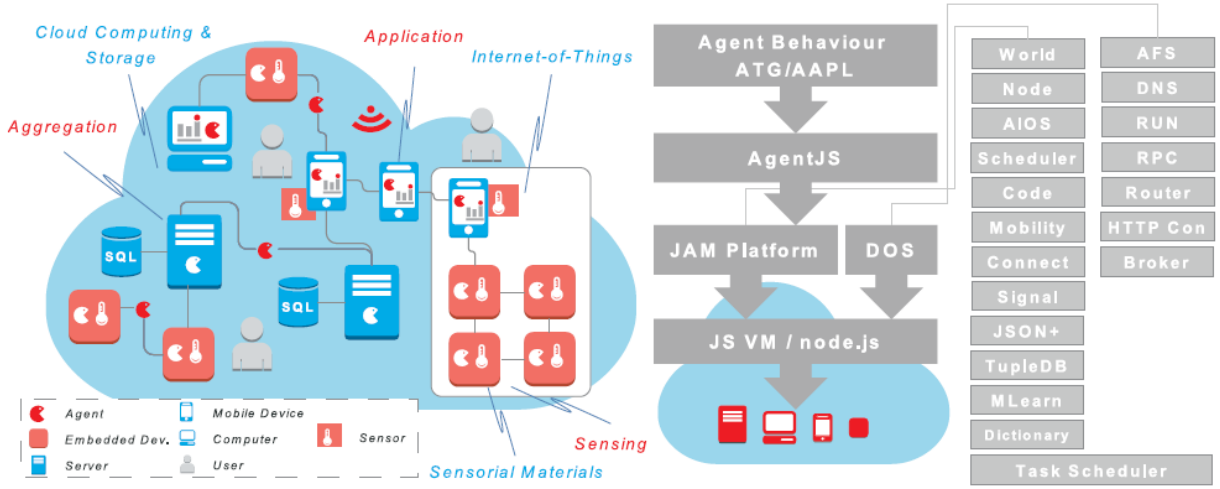


Fig. 1. Unified IoT - Cloud Distributed Perception and Information Processing with mobile agents and a JavaScript (JS) Agent Machine Platform (JAM) and Programming model AgentJS. An optional Distributed Organization System (DOS) layer [7] adds connectivity and security to JAM in the Internet domain.

- *"Agent mobility crossing different execution platforms in networks and agent interaction by using tuple-space databases and global signal propagation aid solving data distribution and synchronization issues in the design of distributed sensor networks"*

Bosse, S. (2016, August). Mobile multi-agent systems for the internet-of-things and clouds using the javascript agent machine platform and machine learning as a service. In *2016 IEEE 4th international conference on future internet of things and cloud (FiCloud)* (pp. 244-253). IEEE.

Tuple-Space

- *"unified agent interaction is provided by using synchronized Linda-like tuple database space access"*
- *Reading bases on template pattern matching and can block agent execution if there is actually no match.*

Example

```
1   agent explorer(dir,radius)
2    var x,y:int;
3    var mean,hop:int;
4    var goback:boolean;
5    activity init = ..
6    end;
7    acitivity move =
8      if (hop=radius) goback := true;
9      else begin hop++; moveto(dir); end;
10   end;
11   activity percept =
12     var s:int; rd(SENSOR,s?);
13     mean := (mean+s)/2;
14   end;
15   activity goback =
16     dir=opposite(dir);
17   end;
18   activity deliver =
19     out(MEAN,mean); signal($parent,DELIVER);
20   end;
21   handler S(v) = .. end;
```

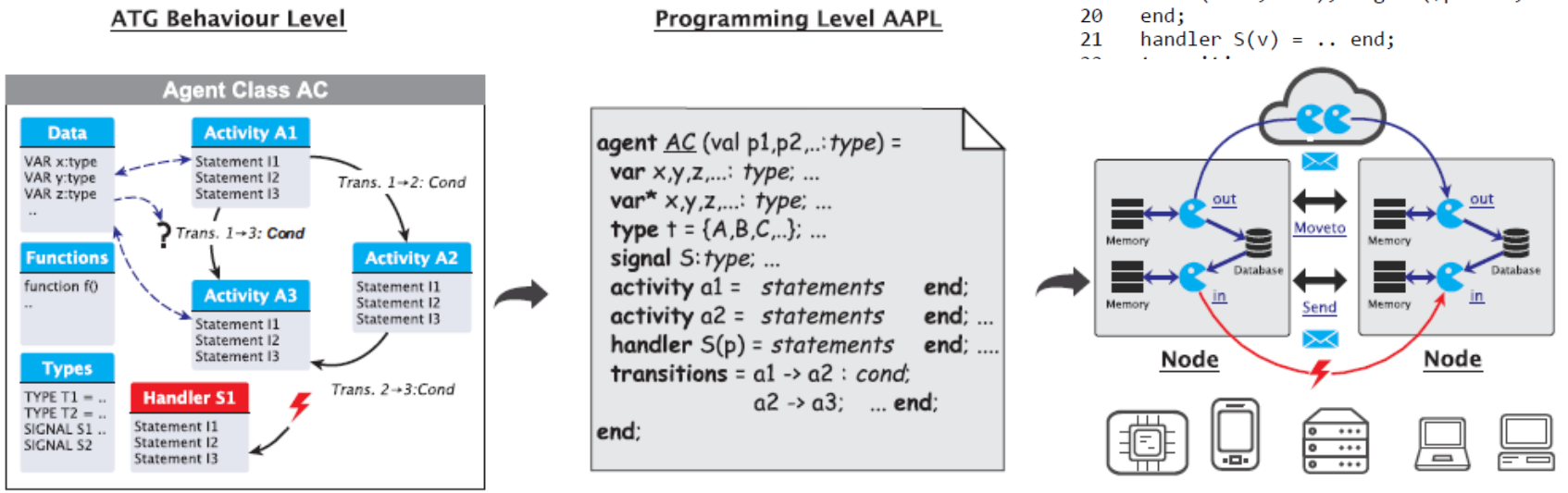


Fig. 2. Agent behaviour programming level with activities and transitions (AAPL, middle); agent class model and activity-transition graphs (left); agent instantiation, processing, and agent interaction on the network node level (right) [12].

Conclusion

- **Environment-based Coordination**
  - Environment definition is broader than the part related to the coordination
  - Environment mediates interaction between agents
  - Environment mediation may be used for coordination with the sharing of information.