Multi-Agent Oriented Programming

– Environment –

O. Boissier

Univ. Lyon, IMT Mines Saint-Etienne, LaHC UMR CNRS 5516, France

CPS2 M1 – Winter 2020

Multi-Agent Oriented Programming

Agent working environment: concepts and approaches

Outline

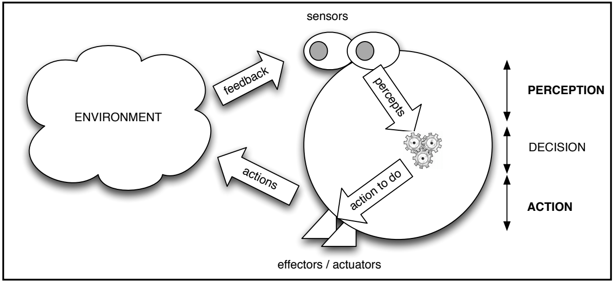
Fundamentals

Existing approaches

Back to the Notion of Environment in MAS

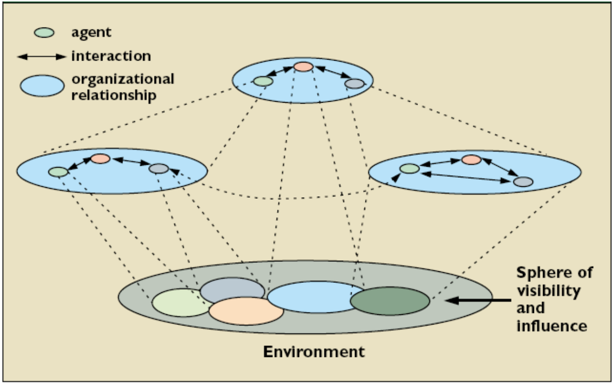
- The notion of environment is intrinsically related to the notion of agent and multi-agent system
  - **"An agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objective"** [Wooldridge, 2002]
  - **"An 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

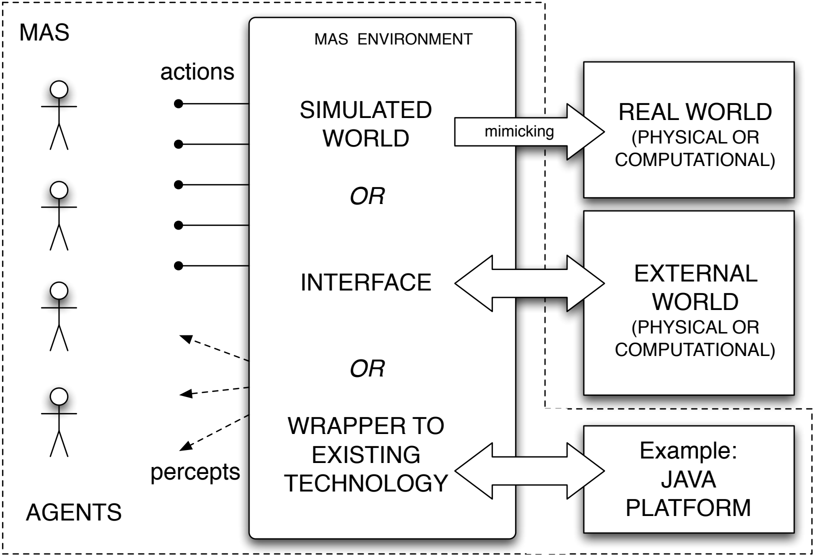
Environment Programming: General Issues

- 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

Outline

Outline

Basic Level Overview

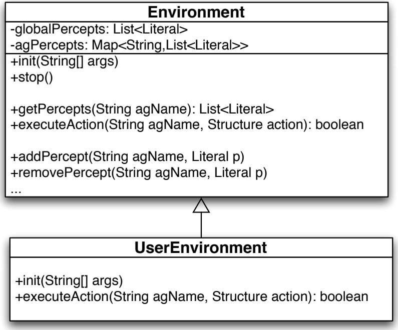


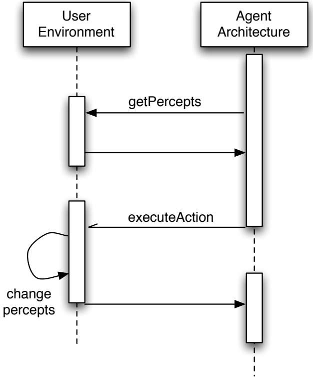
Basic Level: Features

- Environment conceptually conceived as a single monolitic block
  - providing actions, generating percepts
- Environment API
  - to define the set of actions and program actions computational behaviour
    - which include the generation of percepts
  - typically implemented using as single object/class in OO such as Java
    - method to execute actions
    - fields to store the environment state
  - available in many agent programming languages/frameworks
    - e.g., Jason, 2APL, GOAL, JADEX

An Example: *Jason* [Bordini et al., 2007] (without JaCaMo)

- Flexible Java-based Environment API
  - Environment base class to be specialised
    - executeAction method to specify action semantics
    - addPercept to generate percepts





Example (continued): MARS Environment in *Jason*

```
public class MarsEnv extends Environment {
  private MarsModel model;
  private MarsView  view;

  public void init(String[] args) {
        model = new MarsModel();
        view  = new MarsView(model);
        model.setView(view);
        updatePercepts();
  }

  public boolean executeAction(String ag, Structure action) {
    String func = action.getFunctor();
    if (func.equals("next")) {
      model.nextSlot();
    } else if (func.equals("move_towards")) {
      int x = (int)((NumberTerm)action.getTerm(0)).solve();
      int y = (int)((NumberTerm)action.getTerm(1)).solve();
      model.moveTowards(x,y);
    } else if (func.equals("pick")) {
      model.pickGarb();
    } else if (func.equals("drop")) {
      model.dropGarb();
    } else if (func.equals("burn")) {
      model.burnGarb();
    } else {
      return false;
    }

    updatePercepts();
    return true;
  }
  ...
```

```
  ...

    /* creates the agents perception
     * based on the MarsModel */
  void updatePercepts() {

    clearPercepts();

    Location r1Loc = model.getAgPos(0);
    Location r2Loc = model.getAgPos(1);

    Literal pos1 =  Literal.parseLiteral
        ("pos(r1," + r1Loc.x + "," + r1Loc.y + ")");
    Literal pos2 = Literal.parseLiteral
        ("pos(r2," + r2Loc.x + "," + r2Loc.y + ")");

    addPercept(pos1);
    addPercept(pos2);

    if (model.hasGarbage(r1Loc)) {
      addPercept(Literal.parseLiteral("garbage(r1)"));
    }

    if (model.hasGarbage(r2Loc)) {
     addPercept(Literal.parseLiteral("garbage(r2)"));
    }
  }

  class MarsModel extends GridWorldModel { ... }

  class MarsView extends GridWorldView { ... }
}
```

Example (continued): *Jason* Agents Playing on Mars

```
// mars robot 1

/* Initial beliefs */

at(P) :- pos(P,X,Y) & pos(r1,X,Y).

/* Initial goal */

!check(slots).

/* Plans */

+!check(slots) : not garbage(r1)
   <- next(slot);
      !!check(slots).
+!check(slots).

+garbage(r1) : not .desire(carry_to(r2))
   <- !carry_to(r2).

+!carry_to(R)
   <- // remember where to go back
      ?pos(r1,X,Y);
      -+pos(last,X,Y);

      // carry garbage to r2
      !take(garb,R);

      // goes back and continue to check
      !at(last);
      !!check(slots).
...
```

```
...

+!take(S,L) : true
   <- !ensure_pick(S);
      !at(L);
      drop(S).

+!ensure_pick(S) : garbage(r1)
   <- pick(garb);
      !ensure_pick(S).
+!ensure_pick(_).

+!at(L) : at(L).
+!at(L) <- ?pos(L,X,Y);
           move_towards(X,Y);
           !at(L).
```

Another Example: **2APL** [Dastani, 2008]

- 2APL
  - BDI-based agent-oriented programming language integrating declarative programming constructs (beliefs, goals) and imperative style programming constructs (events, plans)
- Java-based Environment API
  - `Environment` base class
  - implementing actions as methods
    - inside action methods external events can be generated to be perceived by agents as percepts

Example: Block-world Environment in **2APL**

```
package blockworld;

public class Env extends apapl.Environment {


  public void enter(String agent, Term x, Term y, Term c){...}

  public Term sensePosition(String agent){...}

  public Term pickup(String agent){...}

  public void north(String agent){...}

  ...

}
```

2APL Agents in the block-world

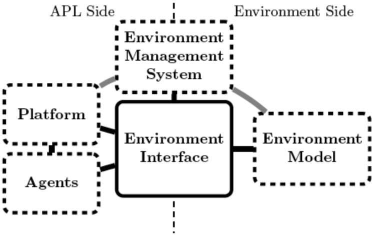
```
BeliefUpdates:
  { bomb(X,Y) }          RemoveBomb(X,Y){ not bomb(X,Y) }
  { true }               AddBomb(X,Y)   { bomb(X,Y) }
  { carry(bomb) }        Drop( )        { not carry(bomb)}
  { not carry(bomb) }    PickUp( )      { carry(bomb) }

Beliefs:
  start(0,1).
  bomb(3,3).
  clean( blockWorld ) :-
     not bomb(X,Y) , not carry(bomb).

Plans:
  B(start(X,Y)) ;
  @blockworld( enter( X, Y, blue ), L )

Goals:
  clean( blockWorld )

PG-rules:
  clean( blockWorld ) <- bomb( X, Y ) |
  {
    goto( X, Y );
    @blockworld( pickup( ), L1 );
    PickUp( );
    RemoveBomb( X, Y );
    goto( 0, 0 );
    @blockworld( drop( ), L2 );
    Drop( )
  }
...
```

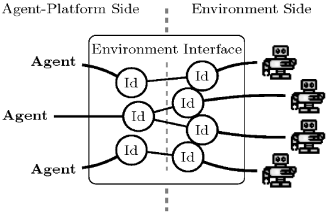
```
...

PC-rules:
  goto( X, Y ) <- true |
  {
    @blockworld( sensePosition(), POS );
    B(POS = [A,B]);
    if B(A > X) then
    { @blockworld( west(), L );
      goto( X, Y )
    }
    else if B(A < X) then
    { @blockworld( east(), L );
      goto( X, Y )
    }
    else if B(B > Y) then
    { @blockworld( north(), L );
      goto( X, Y )
    }
    else if B(B < Y) then
    { @blockworld( south(), L );
      goto( X, Y )
    }
  }

  ...
```

Environment Interface Standard – EIS Initiative

- Recent initiative supported by main APL research groups [Behrens et al., 2010]
  - GOAL, 2APL, GOAL, JADEX, JASON
- Goal of the initiative
  - design and develop a generic environment interface standard
    - a standard to connect agents to environments
    - ... environments such as agent testbeds, commercial applications, video games..
- Principles
  - wrapping already existing environments
  - creating new environments by connecting already existing apps
  - creating new environments from scratch
- Requirements
  - generic
  - reuse

EIS Meta-Model





- By means of the Env. Interface agents perform actions and collect percepts
  - actually actions/percepts are issued to controllable entities in environment model
  - represent the agent bodies, with effectors and sensors

Environment Interface Features

- Interface functions
  - attaching, detaching, and notifying observers (software design pattern);
  - registering and unregistering agents;
  - adding and removing entities;
  - managing the agents-entities-relation;
  - performing actions and retrieving percepts;
  - managing the environment
- Interface Intermediate language
  - to facilitate data-exchange
  - encoding percepts, actions, events

Outline

Advanced Level Overview

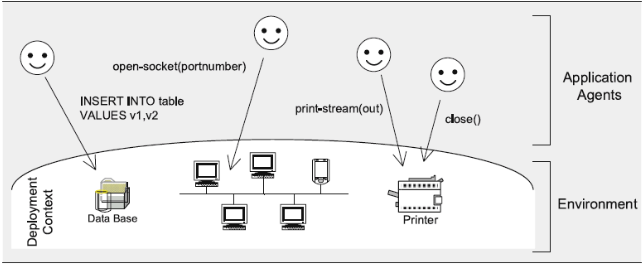
- Vision: environment as a **first-class abstraction** in MAS [Weyns et al., 2007, Ricci et al., 2010]
  - **application** or **endogenous** environments, i.e. that environment which is an explicit part of the MAS
  - providing an exploitable **design** & **programming** abstraction to build MAS applications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

Three Support Levels [Weyns et al., 2007]

- Basic **interface** support
- **Abstraction** support level
- **Interaction-mediation** support level

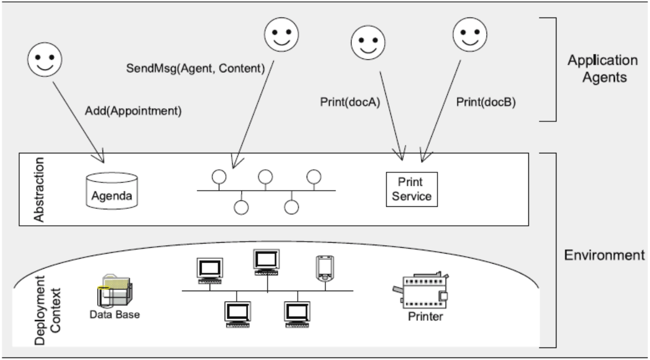
Basic Interface Support

- The environment enables agents to access the deployment context
  - i.e. the hardware and software and external resources with which the MAS interacts



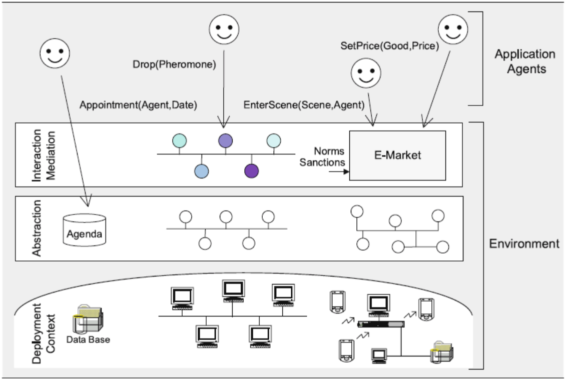
Abstraction Support

- Bridges the conceptual gap between the agent abstraction and low-level details of the deployment context
  - shields low-level details of the deployment context



Interaction-Mediation Support

- **Regulate** the access to shared resources
- **Mediate** interaction between agents



Environment Definition Revised

Environment definition revised [Weyns et al., 2007]

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

Research on Environments for MAS

- Environments for Multi-Agent Systems research field / **E4MAS** workshop series [Weyns et al., 2005]
  - different themes and issues (see JAAMAS Special Issue [Weyns and Parunak, 2007] for a good survey)
    - mechanisms, architectures, infrastructures, applications [Platon et al., 2007, Weyns and Holvoet, 2007, Weyns and Holvoet, 2004, Viroli et al., 2007]
  - the main perspective is (agent-oriented) software engineering
- In MAOP, role of the environment abstraction in **MAS programming**
  - **environment programming**

Environment Programming

- Environment as **first-class programming abstraction** [Ricci et al., 2010]
  - software designers and engineers perspective
  - **endogenous** environments (vs. exogenous one)
  - programming MAS = programming Agents + programming Environment
    - ..but this will be extended to include OOP in next part
- Environment as **first-class runtime abstraction** for agents
  - agent perspective
  - to be observed, used, adapted, constructed, ...
- Defining computational and programming frameworks/models also for the environment part

Computational Frameworks for Environment Programming: Issues

- Defining the environment interface
  - actions, percepts, data model
  - **contract** concept, as defined in software engineering contexts (Design by Contract)
- Defining the environment computational model
  - environment structure, behaviour
- Defining the environment distribution model
  - topology

Programming Models for the Environment: Desiderata

- **Abstraction**
  - keeping the agent abstraction level e.g. no agents sharing and calling OO objects
  - effective programming models for controllable and observable computational entities
- **Modularity**
  - away from the monolithic and centralised view
- **Orthogonality**
  - wrt agent models, architectures, platforms
  - support for heterogeneous systems
- **Dynamic extensibility**
  - dynamic construction, replacement, extension of environment parts
  - support for open systems
- **Reusability**
  - reuse of environment parts for different kinds of applications

Existing Computational Frameworks

- AGRE / AGREEN / MASQ [Stratulat et al., 2009]
  - AGRE – integrating the AGR (Agent-Group-Role) organisation model with a notion of environment
    - Environment used to represent both the physical and social part of interaction
  - AGREEN / MASQ – extending AGRE towards a unified representation for physical, social and institutional environments
  - Based on MadKit platform [Gutknecht and Ferber, 2000]
- GOLEM [Bromuri and Stathis, 2008]
  - Logic-based framework to represent environments for situated cognitive agents
  - composite structure containing the interaction between cognitive agents and objects
- A&A and CArtAgO [Ricci et al., 2010]
  - introducing a computational notion of artifact to design and implement agent environments

Bibliography I

Behrens, T., Bordini, R., Braubach, L., Dastani, M., Dix, J., Hindriks, K., Hübner, J., and Pokahr, A. (2010).
An interface for agent-environment interaction.
In *In Proceedings of International Workshop on Programming Multi-Agent Systems (ProMAS-8)*.

Bordini, R., Hübner, J., and Wooldridge, M. (2007).
Programming Multi-Agent Systems in AgentSpeak Using Jason.
Wiley-Interscience.

Bromuri, S. and Stathis, K. (2008).
Situating Cognitive Agents in GOLEM.
In Weyns, D., Brueckner, S., and Demazeau, Y., editors, *Engineering Environment-Mediated Multi-Agent Systems*, volume 5049 of *LNCS*, pages 115–134. Springer Berlin / Heidelberg.

Dastani, M. (2008).
2APL: a practical agent programming language.
Autonomous Agent and Multi-Agent Systems, 16(3):214–248.

Bibliography II

Gutknecht, O. and Ferber, J. (2000).
The MADKIT agent platform architecture.
In *Agents Workshop on Infrastructure for Multi-Agent Systems*, pages 48–55.

Platon, E., Mamei, M., Sabouret, N., Honiden, S., and Parunak, H. V. (2007).
Mechanisms for environments in multi-agent systems: Survey and opportunities.
Autonomous Agents and Multi-Agent Systems, 14(1):31–47.

Ricci, A., Piunti, M., and Viroli, M. (2010).
Environment programming in multi-agent systems – an artifact-based perspective.
Autonomous Agents and Multi-Agent Systems.
Published Online with ISSN 1573-7454 (will appear with ISSN 1387-2532).

Russell, S. and Norvig, P. (2003).
Artificial Intelligence, A Modern Approach (2nd ed.).
Prentice Hall.

Bibliography III

Stratulat, T., Ferber, J., and Tranier, J. (2009).
MASQ: towards an integral approach to interaction.
In *AAMAS (2)*, pages 813–820.

Viroli, M., Holvoet, T., Ricci, A., Schelfthout, K., and Zambonelli, F. (2007).
Infrastructures for the environment of multiagent systems.
Autonomous Agents and Multi-Agent Systems, 14(1):49–60.

Weyns, D. and Holvoet, T. (2004).
A formal model for situated multi-agent systems.
Fundamenta Informaticae, 63(2-3):125–158.

Weyns, D. and Holvoet, T. (2007).
A reference architecture for situated multiagent systems.
In *Environments for Multiagent Systems III*, volume 4389 of *LNCS*, pages 1–40. Springer Berlin / Heidelberg.

Weyns, D., Omicini, A., and Odell, J. J. (2007).
Environment as a first-class abstraction in multi-agent systems.
Autonomous Agents and Multi-Agent Systems, 14(1):5–30.

Bibliography IV

Weyns, D. and Parunak, H. V. D., editors (2007).
Special Issue on Environments for Multi-Agent Systems, volume 14 (1) of *Autonomous Agents and Multi-Agent Systems*. Springer Netherlands.

Weyns, D., Parunak, H. V. D., Michel, F., Holvoet, T., and Ferber, J. (2005).
Environments for multiagent systems: State-of-the-art and research challenges.
In Weyns, D., Parunak, H. V. D., Michel, F., Holvoet, T., and Ferber, J., editors, *Environment for Multi-Agent Systems*, volume 3374, pages 1–47. Springer Berlin / Heidelberg.

Wooldridge, M. (2002).
An Introduction to Multi-Agent Systems.
John Wiley & Sons, Ltd.