

**USD**

---

**Odense  
09 August 2010**

## **MAS Course 4**

**Yves Demazeau**  
Yves.Demazeau@imag.fr

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 197

### **SCHEDULE OF THE COURSE + EXAMINATION**

---

<b>MAS 01</b>	<b>04 Aug.</b>	<b>Introduction, Methodology, Agents</b>
<b>MAS 02</b>	<b>05 Aug.</b>	<b>Agents, Environments, Interactions</b>
<b>MAS 03</b>	<b>06 Aug.</b>	<b>Dynamics, Organisations, Example</b>
<b>MAS 04</b>	<b>09 Aug.</b>	<b>Development, Deployment, Example</b>
<b>MAS 05</b>	<b>10 Aug.</b>	
<b>MAS 06</b>	<b>11 Aug.</b>	

attendance ; handouts ; individual work  
[ Ferber 95 ] [ HERMES 01 ] [ OFTA 04 ]

**MAS Ex.    13 Aug.    Written Control and Oral Delivery**

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 198

## DEVELOPMENT

---

## Evolution of Programming Paradigms

---

### **1950's**

- Machine and assembly language

### **1960's**

- Procedural programming

### **1970's**

- Structured programming

### **1980's**

- Object-Based programming, Declarative programming

### **1990's**

- Frameworks, design patterns, scenarios, and protocols

### **2000's**

- Agents... Multi-Agent Systems...

...

## Features of Languages and Paradigms

Concept	Proc. L.	Object L.	Agent L.
<b>abstraction block</b>	type data	class object	society agent
<b>model</b>	procedure call	method message	perceive reason / act
<b>paradigm</b>	tree of procedures	interaction patterns	cooperative interaction
<b>architecture</b>	functional decomposition	inheritance polymorphism	managers assistants, peers
<b>modes of terminology</b>	coding implement	designing and using engineer	enabling and enacting activate

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 201

## Agent Oriented Programming [Shoham 93]

### A complete AOP system will include three primary components

- a restricted formal language with clear syntax and semantics for describing mental state: the mental state will be defined uniquely by several modalities, such as belief and commitment
- an interpreted programming language in which to define and program agents, with primitive commands such as REQUEST and INFORM: the semantics of the language will be required to be faithful to the semantics of the mental state
- an "agentifier", converting neutral devices into programmable agents.

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 202

## **AOP vs. OOP : introduction**

- The use of mental states is to design the computational system
- From an engineering point of view, AOP can be viewed as a specialization of OOP, in the original sense of Hewitt's Actors model
- OOP proposes viewing a computational system as made up of modules that are able to communicate with one another and that have individual ways of handling incoming messages
- AOP specializes the framework by fixing the state (mental state) of the modules (agents) to consist of components such as beliefs (about the world, about themselves, about one another), capabilities, and decisions, each of which enjoys a precisely defined syntax.

## **Components of mental state**

### **Components of mental state**

- the future is determined by two factors : past history and current actions of agents.

### **The actions of an agent are determined by its decisions, or choices**

### **Decisions are logically constrained, though not determined, by the agent's beliefs**

- These beliefs refer to the state of the world, to the mental state of other agents, and to the capabilities of this and other agents

### **Basics : beliefs, capabilities, obligations (decision is simply an obligation to oneself)**

## A language for belief, obligation, and capabilities

time / action	$\text{holding}(\text{robot}, \text{cup})^t$
belief	$B_a^t \varphi$ $\varphi$ a (recursive) sentence
obligation	$OBL_{a,b}^t \varphi$
decision	$\equiv_{\text{def}} OBL_{a,a}^t \varphi$
capability	$CAN_a^t \varphi$  $ABLE_a^t \varphi =_{\text{def}} CAN_{\text{time}(\varphi)_a}^t \varphi$ time( $\varphi$ ) the outermost time occurring on

## Properties of the various components

internal consistency	$\forall a, t \{ \varphi : B_a^t \varphi \}$ is consistent $\forall a, t \{ \varphi : OBL_{a,b}^t \varphi \}$ is consistent
good faith	$\forall t, a, b, \varphi \ OBL_{a,b}^t \varphi \supset B_a^t ((ABLE_a^t \varphi) \wedge \varphi)$
introspection	$\forall t, a, b, \varphi \ OBL_{a,b}^t \varphi \equiv B_a^t OBL_{a,b}^t \varphi$ $\forall t, a, b, \varphi \neg OBL_{a,b}^t \varphi \equiv B_a^t \neg OBL_{a,b}^t \varphi$
persistence	so are mental state, obligations
capability	capabilities do not fluctuate widely

## Agent0 : basic loop

**The role of agent programs is to control the evolution of an agent's mental state**

- actions occur as a side-effect of the agent's being committed to an action whose time has come

**Each agent iterates the following two steps at regular intervals**

- read current messages, and update your mental state, including your beliefs and commitments (the agent program is crucial for this update)
- execute the commitments for the current time, possibly resulting in further belief change (this phase is independent of the agent's program)

## Agent0 : syntax

**fact statements atomic objective sentences**

**variables**                      ?x

<b>private statements</b>	(DO t p-action)
<b>communicative st.</b>	(INFORM t a fact)
	(REQUEST t a action)
	(UNREQUEST t a action)
	(REFRAIN action)
<b>conditional st.</b>	(IF mntlcond action)
	mental conditions may contain
	logical connectives
<b>commitment rules</b>	(COMMIT msgcond mntlcond
	(agent action) * )

## **Agent0 : interpreter (1)**

**Since it is an instance of the generic interpreter, the AGENT-0 interpreter inherits its two-step loop design**

**The first loop may be specialized as follows**

- update the beliefs : Agent0 imposes an extreme restriction, which is to disallow logical connectives other than negation. This makes the consistency checking trivial, at most linear in the size of the database.
- update the commitments : existing commitments are removed either as a result of the belief change, or as a result of UNREQUEST messages.

## **Agent0 : interpreter (2)**

**Adding commitments is performed as follows**

**for each program statement  
(COMMIT msgcond mntlcond ( $a_i$  action $_i$ ) \*)**

**if**

- msgcond holds the new incoming message,
- mntlcond holds the current mental state
- $\forall i$ ,  $a_i$  is currently capable of the action $_i$ , and
- $\forall i$ ,  $a_i$  is not committed to REFRAIN action $_i$ , and, if action $_i$  is itself of the form REFRAIN action $_i$ , the agent is not committed to action $_i$ ,

**then**

- $\forall i$ , commit to  $a_i$  to perform action $_i$

## **Interaction Oriented Programming [Huhns 96]**

### **Motivations**

- errors will always be in complex systems;
- Error-free code can be a disadvantage;
- Where systems interact with the real world, there is a power that can be exploited

### **Example : children forming a circle**

- conventional approach: create a C++ class for each type of object, write a control program that uses trigonometry to compute the location of each object
- interaction-oriented approach: children approach is robust due to local intelligence and autonomy, write the program based on objects having attitudes, goals, agent models

**IOP : Active modules, declarative specification, modules that volunteer, modules hold belief about the world, especially about themselves and others**

## **Organisation Oriented Programming [Lemaitre 98]**

**Designing, Maintaining, Using MAS utilize different integrative frameworks that include features to deal with agents, interactions, environments, ... MAS programming itself follows history of programming.**

**The most well-known effort towards MAOP is AOP [Shoham 93] ... IOP [Huhns 97] is an alternative...**

**OOP is another one [Lemaitre 98] ... EOP does not actually exist as a trend but looks like Artificial Life.**

**These approach respectively focus on Agents, on Interactions, on Organisations, on Environments, as being the respective basic bricks at the disposal of the designer / MAS / user...**



### **((A + I) + O) + E) OP : Populations [Demazeau 96]**

**The Population structure is the set of agents, the set of possible behaviors of the agents, and the set of all interaction processes between agents**

**Pop = (Ag, Bh, Ip; bc, ic)**

**Ag : set of agents**

**Bh : set of behaviors agents are able to perform**

**Ip : set of interaction processes**

**bc : Ag  $\rightarrow$  P(Bh), behavioral capability,  
bc(a), set of behavior a is able to perform**

**ic : Ag x Ag  $\rightarrow$  P(Ip), interaction capability,  
ic(a1,a2), set of interaction processes  
agents a1 and a2 may perform together**

### **((A + I) + O) + E) OP : Organisations [Demazeau 96]**

**The Organization structure is composed of organizational roles and organizational links**

**Org = (Ro; Li)**

**Ro is defined in a relational way**

- e.g.  $Ro \subseteq Lp \times Gp$  : global processes (Gp) and local processes (Lp), the role is the part of agent's behavior that is integrated in the global process.
- e.g.  $Ro \subseteq Fo \times Lv$  : foci of interest (Fo), representation levels (Lv), the role is the agent's behavior for a given focus at a given level.

**$Li \subseteq Ro \times Ro$**

**(((A + I) + O) + E) OP : Pop  $\bowtie$  Org [Demazeau 96]**

**The suitable relation between the Pop and the Org is the system's organization implementation**

**It is any relation  $\text{imp} = \text{Pop} \bowtie \text{Org}$ , on  $(\text{Ro} \times \text{Ag}) \cup (\text{Li} \times \text{Ip})$ ,  $\text{Pop} = (\text{Ag}, \text{Bh}, \text{Ip}; \text{bc}, \text{ic})$ ,  $\text{Org} = (\text{Ro}; \text{Li})$ .**

- if  $(r, a) \in \text{imp}$ ,  $r$  is said to implemented by  $a$
- if  $(l, p) \in \text{imp}$ ,  $l$  is said to implemented by  $p$

**$\text{imp}$  is said "proper" iff  $\bowtie$  is an homomorphism.**

- $\forall r \in \text{Ro}, \exists a \in \text{Ag} / (r, a) \in \text{imp}$ , and  $r$  is properly implemented by some behavior  $b \in \text{bc}(a)$
- $\forall l = (l_1, l_2) \in \text{Li}, \exists ip \in \text{Ip} / \{ (l, ip) \in \text{imp} \wedge \exists (a_1, a_2) \in \text{Ag} \times \text{Ag} / ip \in \text{ic}(a_1, a_2), (r_1, a_1) \in \text{imp}, (r_2, a_2) \in \text{imp}, \text{ and } r_1, r_2 \text{ are properly implemented by the behaviors of } a_1 \text{ and } a_2, \text{ respectively } \}$

**(((A + I) + O) + E) OP : PopOrgs [Demazeau 96]**

**The Interior ( = Population + Organisation ) of a time-invariant multi-agent system is captured by a population-organization structure  $\text{PopOrg} = (\text{Pop}, \text{Org}; \text{imp})$ , where**

- $\text{Pop} = (\text{Ag}, \text{Bh}, \text{Ip}; \text{bc}, \text{ic})$  is a population structure
- $\text{Org} = (\text{Ro}; \text{Li})$  is a organization structure
- $\text{imp} \subseteq (\text{Ro} \times \text{Ag}) \cup (\text{Li} \times \text{Ip})$  is an organization implementation relation as defined previously

## Changes in population and organization structures

### The three main kinds of changes in the population structure $\text{Pop} = (\text{Ag}, \text{Bh}, \text{Ip}; \text{bc}, \text{ic})$

- change of  $\text{bc}(a)$  of some agent  $a$
- change in  $\text{ic}(a1, a2)$  of some agents  $(a1, a2)$
- change in the population set  $\text{Ag}$

### The four main kinds of changes in the organization structure $\text{Org} = (\text{Ro}; \text{Li})$

- change in a role  $r$  from  $\text{Ro}$
- change in a link  $l$  from  $\text{Li}$
- change in the set of roles  $\text{Ro}$
- change in the set of roles  $\text{Li}$

## Multi-Agent Oriented Programming

### Not Object-Oriented Programming

- $S = \text{Objects} + \text{Message passing}$

### Not Logic nor Expert Systems Programming

- $S = \text{Knowledge} + \text{Inference Mechanism}$

### Not Ontology-Oriented Programming

- $S = \text{Knowledge} + \text{Problem Solving Methods}$

### But Agent-Oriented Programming

- $S = \text{BDI Agents} + \text{KQML (Interactions)}$

### But $((A + I) + O) + E$ -Oriented Programming

- $S = ((A + I) + O) + E$

### But VOWELS Programming

- $S = [A^*; E^*; I^*; O^*] + (\text{Recursion \& Emergence}) \text{ Mechanism}$

## The AEIO toolboxes of the MAGMA approach

### Agents

- internal architectures of the processing entities

### Environment

- domain-dependent elements for structuring external interactions between entities

### Interactions

- elements for structuring internal interactions between agents

### Organizations

- elements for structuring sets of agents within the multi-agent system

**These definitions are agent-centered but alternatives exist, firstly based on environments, on interactions, or on organisations.**

## Systems structuring in the MAGMA approach

### The Declarative Principle

$$\text{MAS} = \text{A} + \text{E} + \text{I} + \text{O}$$

### The Functional Principle

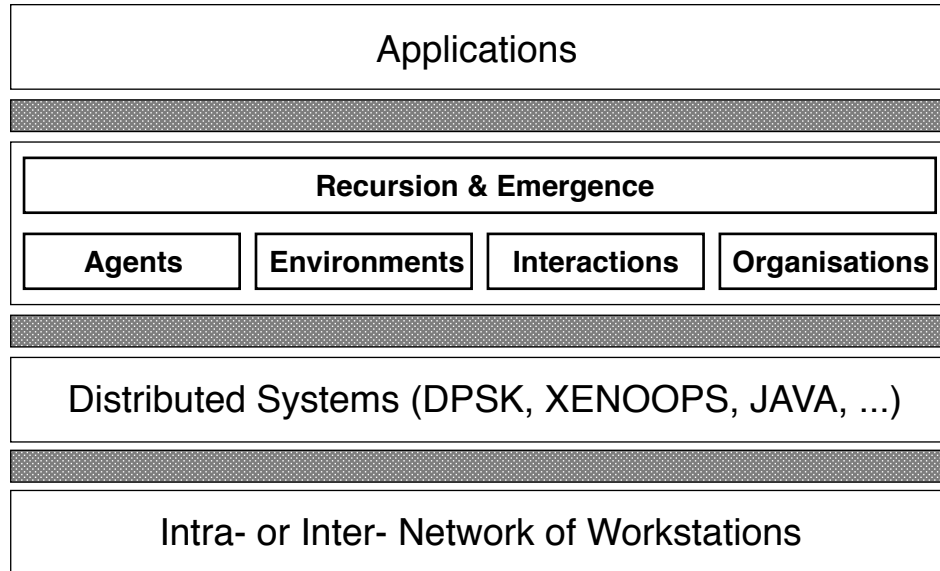
$$\text{Function}(\text{MAS}) = \sum \text{Function}(\text{A}) \\ + \text{Emergence Function}$$

### The Recursive Principle

$$\text{A} = \text{basic A} \mid \text{MAS}$$

**These principles are agent-centered but alternatives exist : any recursion dealing with A, E, I, O or any combination of the four basic elements**

## Programming with the MAGMA approach



## MAOP and Domain Orientation (1)

$((A + E) + I) + O$       **Robotics Science**

$((A + I) + O) + E$       **Social Science**

$((E + A) + (I + O))$       **Life Science**

$((I + O) + A) + E$       **Military Science**

$((O + I) + E) + A$       **Economic Science**

**The purpose of the domain drives the design of the system**

## MAOP and Domain Orientation (2)

**MAS are not always A-centered !**

$((A + E) + I) + O$	Robotics Science
$((A + I) + O) + E$	Social Science
$((E + A) + (I + O))$	Life Science
$((I + O) + A) + E$	Military Science
$((O + I) + E) + A$	Economic Science

**The purpose of the domain drives the design of the system**

## MAOP and Domain Orientation (3)

PhD Boissier	$(A + I) + O$	ASIC
PhD Sichman	$A + O$	DEPNET
PhD Ferrand	$((A + I) + O) + E$	SANPA
PhD Baeijs	$((A + E) + I) + O$	SIGMA
PhD Van Aeken	$O + A$	SMAMS
PhD Ribeiro	$I + A$	DIM
PhD Ricordel	Development	VOLCANO

**The purpose of the domain drives the design of the system...**

## The Domain Orientation and the User Orientation

PhD Boissier	$(A + I) + O$	ASIC
PhD Sichman	$A + O$	DEPNET
PhD Ferrand	$((A + I) + O) + E$	SANPA
PhD Baeijs	$((A + E) + I) + O$	SIGMA
PhD Van Aeken	$O + A$	SMAMS
PhD Ribeiro	$I + A$	DIM
PhD Ricordel	Development	VOLCANO

The purpose of the domain drives the design of the system...

But how far the user does the user drive the design of the system ? In fact, in all this, where is the User ?

## MAOP and User Orientation (1)

The traditional place of the « User » is the « End- »

$((((A + E) + I) + O) + U)$	Robotics Science
$((((A + I) + O) + E) + U)$	Social Science
$((((E + A) + (I + O)) + U)$	Life Science
$((((I + O) + A) + E) + U)$	Military Science
$((((O + I) + E) + A) + U)$	Economic Science

## MAOP and User Orientation (2)

In fact, in all this, where is the User ? With the A ?

$((((U + A) + E) + I) + O)$       Robotics Science

$((((U + A) + I) + O) + E)$       Social Science

$((E + (U + A)) + (I + O))$       Life Science

$((I + O) + (U + A)) + E$       Military Science

$((O + I) + E) + (U + A)$       Economic Science

But in fact MAS are not always  $((U + A)$ -centered !

## Interactive Games

### General

- ---> A to be replaced by  $(U + A)$

### Handling

- The goal is to master emergence, to optimize a cost
- The understanding of the whole system has to be easy
- The (E)nvironment has to be as realistic as possible
- ---> E to be replaced by  $(U + E)$

### Strategy

- The goal is to use and to plan the use of resources
- The visualisation of the interactions is highly desirable
- The key parameters of the game are (I)nteractions
- ---> I to be replaced by  $(U + I)$

### Role

- The goal is to increase the competences of the User
- Competences of the characters have to be visualized
- The (O)rganisation constitutes the entry of the game
- ----> O to be replaced by  $(U + O)$



### MAOP and User Orientation (3)

From U as consumer...

$((((A + E) + I) + O) + U)$	Robotics Science
$((((O + I) + E) + (U + A)))$	Economic Science

To U as a partner...

$((((I + O) + (U + A)) + E)$	Military Science
$((E + (U + A)) + (I + O)))$	Life Science
$((((U + A) + I) + O) + E)$	Social Science

Towards U as a creator...

$(((((U + A) + E) + I) + O)$	Robotics Science
$((((U + (O + I)) + E) + A)$	Economic Science

### VOWELS A E I O U Decomposition

#### Agents

- internal architectures of the system processing entities

#### Environment

- domain-dependent elements for structuring external interactions between entities

#### Interactions

- elements for structuring internal interactions between entities

#### Organisations

- elements for structuring sets of entities within the MAS

#### Users

- internal architectures of the end-user processing entities

## VOWELS Oriented Programming

### The Declarative Principle

$$\text{MAS} = \text{A} + \text{E} + \text{I} + \text{O} + \text{U}$$

### The Functional Principle

$$\text{Function}(\text{MAS}) = \sum \text{Function}(\text{entities}) \\ + \text{Emergence Function}$$

### The Recursive Principle

$$\text{entity} = \text{basic entity} \mid \text{MAS}$$

### The Programming Principle

$$\text{MAS} = [\text{A}^*; \text{E}^*; \text{I}^*; \text{O}^*; \text{U}^*] \\ + (\text{Recursion \& Emergence}) \text{ Mechanism}$$

## Vowels Oriented Programming

### MAOP subsumes AOP, IOP, OOP-like OP...

#### We defend an instance of MAOP, the VOWELS framework in which :

- 1/ to express the problem to solve independently of the domain
- 2/ to "vowellify" the problem in terms of A E I O U, ...
- 3/ to choose understood frames of A, E, I, O, U, dynamics, and recursion
- 4/ to leave VOWELS "emergence engine" complete the missing bricks by itself and build the appropriate MAS...
- 5/ ... to be deployed as self on a distributed settling...
- 6/ ... to be settled and used interactively

### **The orientation of the domain and of the user**

PhD Boissier	$(A + I) + O$	ASIC
PhD Sichman	$A + O$	DEPNET
PhD Ferrand	$((A + I) + O) + E$	SANPA
PhD Baeijs	$((A + E) + I) + O$	SIGMA
PhD Van Aeken	$O + A$	SMAMS
PhD Ribeiro	$I + A$	DIM
PhD Ricordel	Development	VOLCANO
PhD Tavares	Planning	
PhD Deguet	Emergence	
PhD Piolle	$((U + I) + O) + E$	PAW
PhD Joumaa	Evaluation	MASPAJE
PhD Crepin	$((U + O) + I) + E$	HIPPO
PhD Lacomme	Dynamics	

**The purpose of the user drives the design of the system...**

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 233

### **How MAS Methodology is specific ?**

**= Approach + Model + Tools + Problem + Domain**  
**= Analysis + Design + Development + Deployment**

**It provides a new analysis and design approach**

**It is supported by existing formalisms,**

**It integrates existing programming paradigms,**

**...**

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 234

## DEPLOYMENT

---

## Object-Based (Concurrent) Languages

---

### Actors-like Languages

- Massive Parallelism
- Every interaction is made by passing of asynchronous buffered messages
- Use of the local continuity
- Object-oriented design
- Full distributed control

### Actors [Hewitt]

- An actor is composed of acquaintances, scripts, ...
- A message is an actor containing method, continuation, and complaint actors
- A method is a set of actions : create a new actor, send a message, modify actor's behaviour

## Examples of OBCL

### **PLASMA** [Hewitt, Sallé]

- the first actor language

### **ORIENT-84/K** [Tokoro]

- integrates logics, objects, and control monitor

### **MACE** [Gasser]

- actor-based / large-grain

### **KNOS** [Tsichritzis]

- persistent, mobile objects, dynamic acquisition of methods

### **MERING IV** [Carle, Ferber]

- actor-based reflective language / multi-grain
- multiple metaobjects and circular interpreters

### **ACTALK** [Briot]

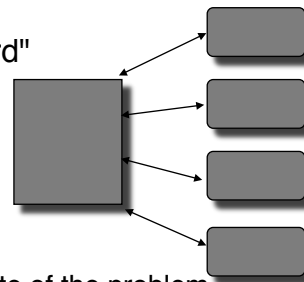
- actors in Smalltalk
- model view controller user interface

...

## (Distributed) Blackboard Systems

### **BB-Like frameworks**

- Interactions through the "blackboard"
- Opportunistic activation of the knowledge sources
- Initially Centralized Control



### **DBB**

- Blackboard
  - ✓ A global database representing state of the problem
  - ✓ Hierarchically organised into levels of granularity
- Knowledge Sources
  - ✓ Produce changes (hypotheses) onto given levels of BB
  - ✓ Contain particular subset of domain knowledge
- Centralized Control
  - ✓ Evaluates the status of the problem
  - ✓ Controls the knowledge sources

## Examples of Blackboard Systems

### **GBB [Corkill]**

- flexible control regimes / control blackboard
- multi-level problem spaces

### **BB1/ BB\* [Hayes-Roth]**

- flexible metalevel control / control blackboard

### **CAGE/POLIGON [Nii]**

- parallel at numerous levels

### **ATOME [Laasri]**

- hierarchical control

### **TRANSACTIONAL B-BOARDS [Ensor, Gabbe]**

- database consistency controls and transaction management integrated with parallel blackboards

...

## Characteristics of the Integrative Environments

### **Language for constructing agents**

- Different agent architectures
- Language for knowledge representation
- Mechanisms for reasoning, deciding, controlling

### **Representing and dealing with the environment**

- Representation of the environment and its evolution
- Implementing the perception of the environment
- Implementing the actions of the agents in the environment

### **Representing and dealing with other agents**

- Primitives, protocols, message processing
- Implementing communication between agents

### **Development Interface**

- Visualisation, trace, inputs-outputs

### **Application Interface**

### **Interface with Target (Distributed or not) System**

## Examples of Integrative Environments

### Cognitive MAS

- **ABE** [Erman]
- **AOP** [Shoham]
- **MASK** [Demazeau et al.]
- **MECCA** [Steiner]
- **MICE** [Durfee]
- **PACT** [Cutkosky]
- **PHOENIX** [Cohen]
- ...

### Reactive MAS

- **ECO** [Drogoul]
- **LYDIA** [Connah]
- **PACO** [Demazeau et al.]
- **RDL** [Steels]
- ...

## MICE [Durfee]

### Michigan Intelligent Coordination Experiment

#### Describe environment in which agents act and interact

- 2-D grid model of the world
- does not restrict agent implementation

#### Variety of organisation structures and cooperation methods built-in

- Contract Net, Greedy Search, Game Theory

#### Pursuit Problem

**ftp'able** (freebie.engin.umich.edu: /pub/Mice)

- Built on Allegro Common Lisp

## **PACT [Cutkosky]**

---

### **Palo Alto Collaborative Testbed**

#### **Computer-Aided Concurrent Engineering**

- Stanford, Lockheed, Hewlett-Packard, Enterprise Integration Technologies
- Have integrated four pre-existing CE systems into common distributed framework supporting aspects of small robot

#### **Uses quasi-standards**

- Knowledge Interchange Format (KIF) [DARPA-92]
- Knowledge Query & Manipulation Language (KQML) [Finin-92]
- Product Data Exchange Specification (PDES)

## **MECCA [Steiner]**

---

### **Multi-Agent Environment for Constructing Cooperative Applications**

**Domain-independent for integrating wide variety of predefined software systems and humans**

#### **Supports agent programming, communication framework, cooperation concepts**

- Interfaces are agents
- MAS editing tools, e.g. monitor, debugger ..., Distributed User Interfaces

#### **Based on ESPRIT Project IMAGINE**

- Implemented on distributed Parlog+Prolog platform



## MASK [Occello]

**Applications** ( .. MAGIC, GEOMED, SMAALA, SIGMA, Le Salon, SPANS.. )  
(Boissier-94)(Ferrand-95)(Baeijs-95)(Van Aeken 96)

### Agents

(Boissier-93)  
(Ferrand-95)  
(Sichman-95)  
(Occello-96)

### Environments

(Demazeau-90)  
(Ferrand-94)  
(Baeijs-95)

### Interactions

(Koning-94)  
(Demazeau-95)  
(Ferrand-96)  
(Pesty-96)

### Organisations

(Baeijs-95)  
(Demazeau-96)  
(Kozlak-96)  
(Van Aeken-96)

**XENOOOPS** (Joosens-94)

**JAVA** (Sun™-95)

Network of Sun-Like Workstations

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 245

## Today's advanced Web offer

### Academics

- Firefly (MIT before Microsoft) (no more accessible)
- MadKit (LIRMM Montpellier - Ferber's group)
- Simula (II Porto Alegre - Alvares's group)
- dMARS (-> Jack, by Agent Oriented Software)
- ...

### Industrials

- Voyager (ObjectSpace) - freeware (linked with OMG)
- JINI (Sun) - freeware
- Aglets (IBM) - freeware
- Javabeans (Sun) - freeware (based on components)
- Agentbuilder (Reticular) - freeware + product (AOP based)
- ZEUS (BT) - freeware product (FIPA compliant)
- ...

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 246

## Qualification criteria

### Four *qualities* for each stages:

- Completeness: quantity & quality
- Applicability: scope, restrictions
- Complexity: competence required, workload
- Reusability: reuse of previous work

### 16 criteria + availability & support

	Analysis	Design	Development	Deployment
Completeness				
Applicability				
Complexity				
Reusability				

## Selected platforms

### Platforms requirements :

- based on a strong academic model
- high quality software, well maintained
- cover as many aspects as possible of MAS
- cover the four methodological stages

### Madkit, Jack, Zeus, AgentBuilder

- Evaluation as of first semester 2000
- Detailed presentation for MadKit (2006)
- Detailed presentation for Jack (2007)

## **MadKit™**

**Developed by O. Gutknecht & J. Ferber, LIRMM**

**Based on the AALAADIN organisational model  
Graphical multi-agent runtime engine**

**Good versatility**

**Light methodology, no BDI**

	Analysis	Design	Development	Deployment
Completeness	none	Aalaadin	Pure Java	G-Box
Applicability	n / a	broad range	simple A	small large MAS
Complexity	n / a	intuitive	few code base	GUI
Reusability	n / a	design patterns	classes	dynamic reconf.

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 249

## **CASSIOPEE : Abstraction Levels**

### **Agents**

- Which architecture to choose to implement the agents ?
- Which scope of knowledge and how to best use it ?
- Which competences and how are they distributed ?

### **Interactions**

- How do agents communicate ?
- Which content ?
- Can agents influence / alterate other's behaviour ?

### **Organisations**

- How do the agents cooperate ?
- Is there a global goal, how to build a plan to reach it ?
- Which structure to organize, which evolution of the structure ?

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 250

## **CASSIOPEE : General Issues**

---

**From the Analysis of natural organisations to the Design of artificial organisations**

**Based on several applications and experiments**

**Three Abstraction Levels**

- individual agents, interactions, organizations

**Agents is defined as a set of Roles**

- individual roles, interactional roles, organizational roles

**Lacks of Models and Tools**

## **The AGR Model: Agent-Group-Role**

---

**Agent An autonomous and communicating entity**

- ✓ The agent plays roles in groups
- ✓ An agent may have different roles and may belong to several groups.

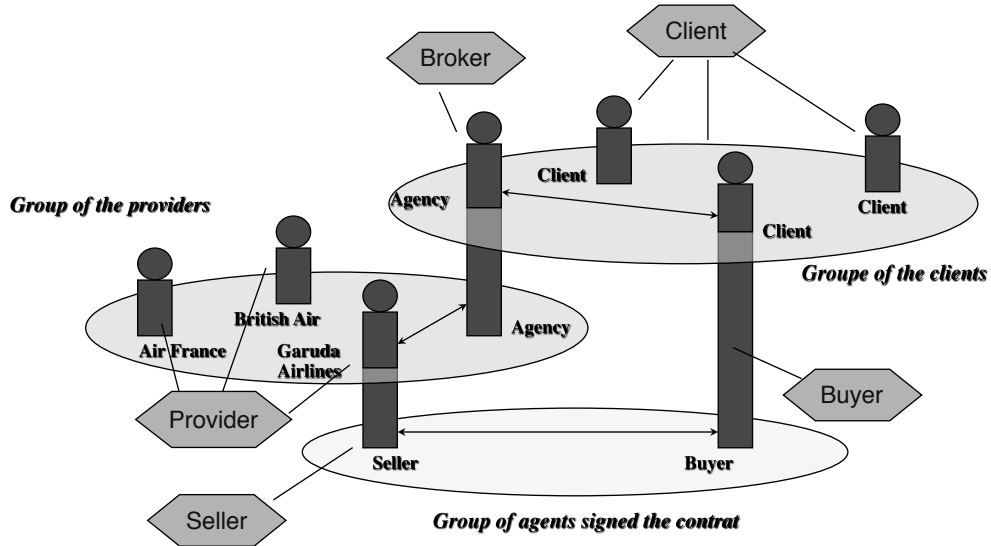
**Group A set of agents sharing a certain characteristics**

- ✓ Two agents can communicate only if they belong to the same group

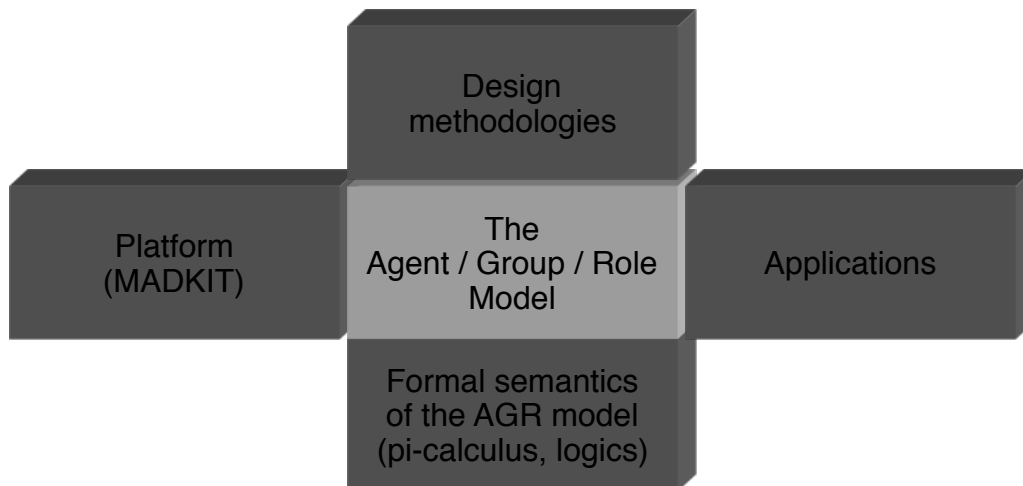
**Role Fonction et point d'interaction**

- ✓ Roles of an agent are local to the groups, and a role has to be requested by the agent.
- ✓ A role can be played by different agents

## The AGR Model: example of an organisation



## The AALAADIN project



## **MadKit™ : Introduction**

**MadKit : meta-platform to develop and execute multi-agent systems**

**The platform is distributed as open source (GPL/LGPL)**

**Applications:**

- Multi-agent simulation and complex systems modelling
- Collective robotics
- « Intelligent » distributed applications

**Several thousands of downloads per year**

**Very used in research and teaching environments**

## **MadKit™ : characteristics**

**Meta-platform to develop and execute multi-agent systems based on the AGR model**

**Does not impose a particular way to program the agents**

- Several programming languages are available: Java, Jess (rule based programming), Python, Scheme

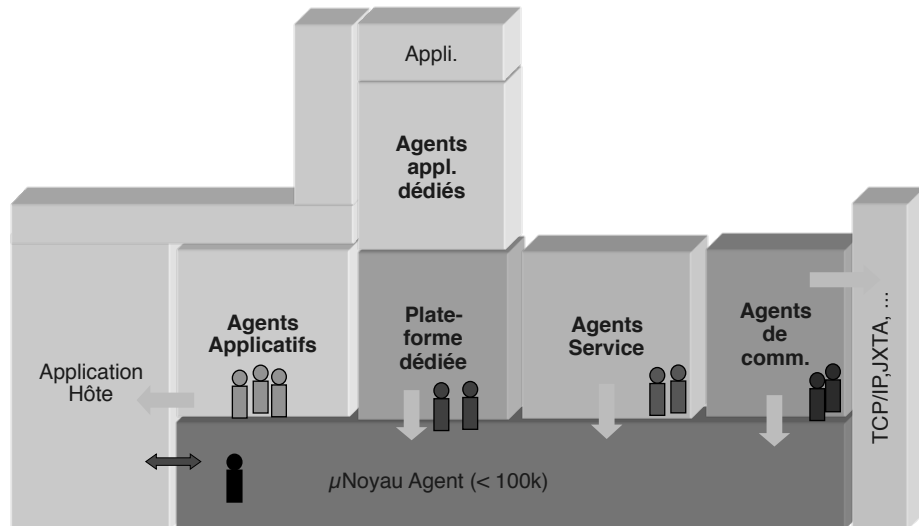
**Enables to have a high heterogeneity of agents (from very ants to more cognitive agents ) running at the same time**

**Peer to peer distribution model of the applications**

- A MadKit application can be directly distributed onto several kernels.
- It enables to make several multi-agent applications run together in a distributed and concurrent way.

**User-friendly interface, including several tools to debug and finalize MAS applications**

## MadKit: General architecture



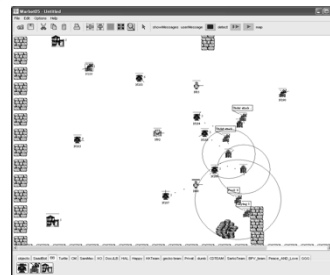
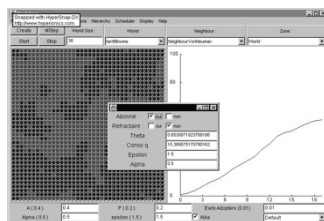
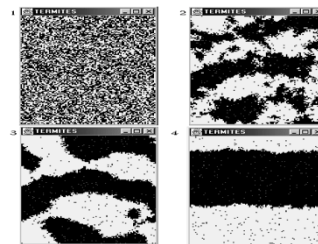
CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 257

## MadKit : meta-platform for development

### Dedicated contexts:

- **TurtleKit**: Simple situated MAS to study emergent phenomena (Lirmm)
- **Warbot**: Study of cooperation principles between agents (Lirmm)
- **Moduleco**: Multi-Agent systems for modeling economical phenomena (Phan, Univ. Brest)



CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 258

## Jack™

**Developed by Agent Oriented Software Pty.**

**Including the dMARS BDI model**

**Great versatility**

**Focus on the development stage**

	Analysis	Design	Development	Deployment
Completeness	none	ident. of classes	extended Java	manual
Applicability	n / a	Jack BDI A	Any MAS	n / a
Complexity	n / a	Jack BDI A	Java & Logic P.	n / a
Reusability	n / a	difficult	classes	n / a

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAIZEAU - 259

## Zeus

**Developed by British Telecom**

**All stages covered, from analysis to deployment**

**Methodological and Software tools**

**Limited to a single agent model**

	Analysis	Design	Development	Deployment
Completeness	role modelling	finding solutions	5 activities	tools docs
Applicability	role o. MAS	task o. A	Zeus A model	debug visualis.
Complexity	UML	design skills	GUI tools	GUI
Reusability	role models	reusable formal.	partial	A reconf.

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAIZEAU - 260



## **AgentBuilder®**

**Developed by Reticular Systems Inc.**

**Grounded on Agent0/Placa BDI architecture**

**Almost all stages covered**

**Complete graphical tools**

**Limited to a single agent model**

	Analysis	Design	Development	Deployment
Completeness	ontology	A definition	behavioural rules	RT A engine
Applicability	universal	cognitive A	AgentBuild. BDI	small societies
Complexity	OO GUI	MAS design GUI	Logic P. GUI	GUI
Reusability	ontology	protocols	A	none

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 261

## **How MAS Methodology is specific ?**

**= Approach + Model + Tools + Problem + Domain**  
**= Analysis + Design + Development + Deployment**

**It provides a new analysis and design approach**

**It is supported by existing formalisms,**

**It integrates existing programming paradigms,**

**It is striving towards industrial quality,**

**...**

CNRS Laboratoire d'Informatique de Grenoble

Yves DEMAZEAU - 262