

# Multi-Agent Oriented Programming

## – Organisation-Oriented Programming –

### The *Moise* Framework

Olivier Boissier

ENS Mines Saint-Etienne  
<http://www.emse.fr/~boissier>

Ecole Nationale  
Supérieure des Mines  
SAINT-ETIENNE

Web Intelligence Master — Nov 2011

Thanks to J.F. Hubner for providing some of the slides

Fundamentals OOP OML OMI E-O A-O Summary **Definition** Motivations

## Intuitive notions of organisation

- ▶ Organisations are structured, patterned systems of activity, knowledge, culture, memory, history, and capabilities that are distinct from any single agent [Gasser, 2001]  
↪ Organisations are **supra-individual** phenomena
- ▶ A decision and communication schema which is applied to a set of actors that together fulfill a set of tasks in order to satisfy goals while guarantying a global coherent state [Malone, 1999]  
↪ definition by the designer, or by actors, to achieve a **purpose**
- ▶ An organisation is characterized by : a division of tasks, a distribution of roles, authority systems, communication systems, contribution-retribution systems [Bernoux, 1985]  
↪ **pattern of predefined cooperation**
- ▶ An arrangement of relationships between components, which results into an entity, a system, that has unknown skills at the level of the individuals [Morin, 1977]  
↪ **pattern of emergent cooperation**



## Outline

- 1 Origins and Fundamentals
- 2 Some OOP approaches
- 3 *MOISE* Organisation Modeling Language (OML)
- 4 *MOISE* Organisation Management Infrastructure (OMI)
- 5 *MOISE* Org. Embodiement Mechanisms for Cartago (E-O)
- 6 *MOISE* Org. Awareness Mechanisms in Jason (A-O)
- 7 Summary

Fundamentals OOP OML OMI E-O A-O Summary **Definition** Motivations

## Organisation in MAS

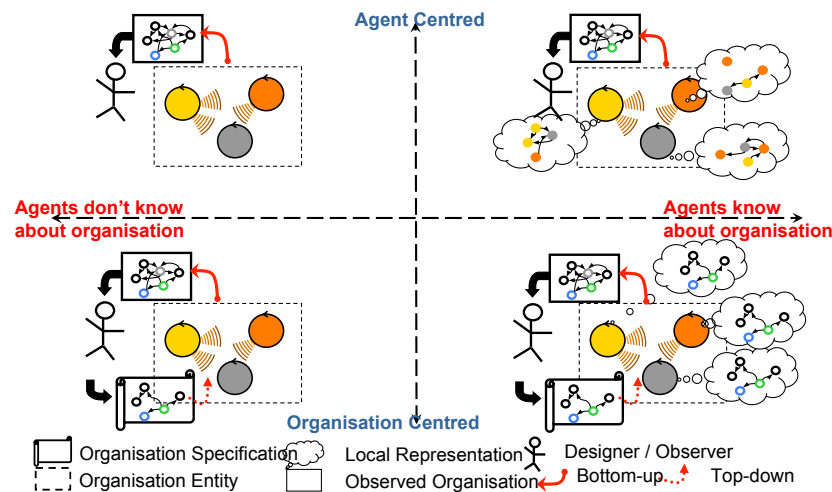
### Definition

Purposive *supra-agent* pattern of emergent or (pre)defined agents cooperation, that could be defined by the designer or by the agents themselves.

- ▶ Pattern of emergent/potential cooperation
  - ▶ called *organisation entity*, institution, social relations, commitments
- ▶ Pattern of (pre)defined cooperation
  - ▶ called *organisation specification*, structure, norms, ...



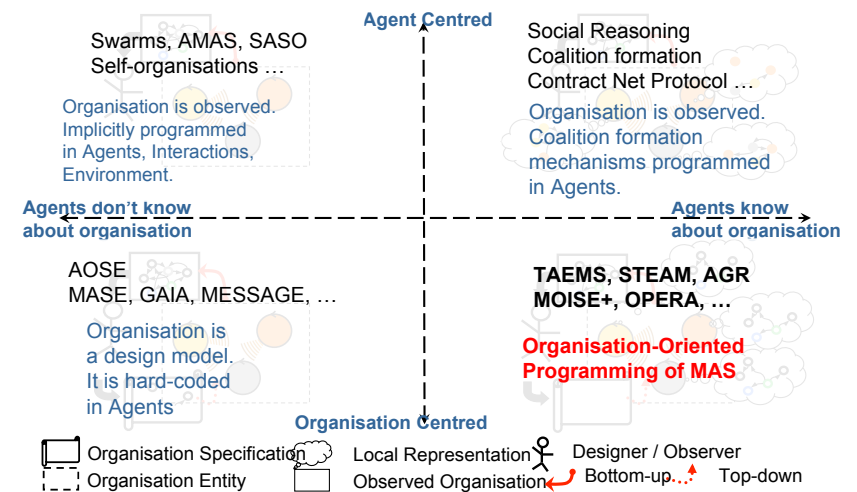
# Perspective on organisations from EASSS'05 Tutorial (Sichman, Boissier)



WI Master, Nov 2011

5 / 107

# Perspective on organisations from EASSS'05 Tutorial (Sichman, Boissier)



WI Master, Nov 2011

6 / 107

## Norms

### Norm

Norms are **rules** that a society has in order to influence the behaviour of agents.

### Norm mechanisms

- ▶ **Regimentation**: norm violation by the agents is prevented  
e.g. the access to computers requires a username  
e.g. messages that do not follow the protocol are discarded
- ▶ **Enforcement**: norm violation by the agents is made possible but it is monitored and subject to incentives  
e.g. a master thesis should be written in two years  
↪ Detection of violations, decision about ways of enforcing the norms (e.g. sanctions)

WI Master, Nov 2011

7 / 107

## Normative Multi-Agent Organisation

### Normative Multi-Agent System [Boella et al., 2008]

A MAS composed of mechanisms to represent, communicate, distribute, detect, create, modify, and enforce norms, and mechanisms to deliberate about norms and detect norm violation and fulfilment.

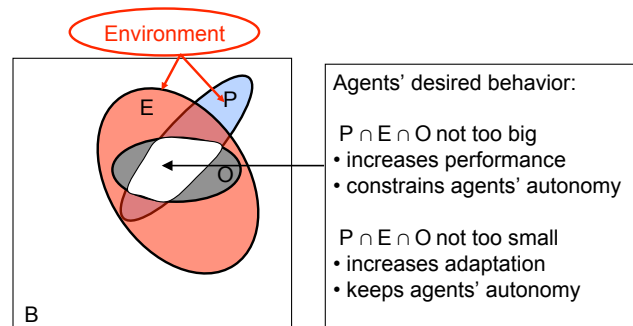
### Normative Multi-Agent Organisation

- ▶ Norms are expressed in the organisation specification:
  - ▶ anchored/situated in the organisation
  - ▶ i.e. norms refer to organisational concepts (roles, groups, )
- ▶ Norms are interpreted and considered in the context of the organisation entity
- ▶ Organisation management mechanisms are complemented with norms management mechanisms (enforcement, regimentation, ...)

WI Master, Nov 2011

8 / 107

## Challenges: Normative Organisation vs Autonomy



- ▶ B: agents possible behaviors
- ▶ P: agents behaviors that lead to global purpose
- ▶ E: agents possible behaviors constrained by the environment
- ▶ O: agents possible/permitted/obliged behaviors constrained by the normative organisation



## Components of OOP: Organisation Modelling Language (OML)

- ▶ Declarative specification of the organisation(s)
- ▶ Specific constraints, norms and cooperation patterns imposed on the agents
  - e.g. AGR [Ferber and Gutknecht, 1998], TEAMCORE [Tambe, 1997], ISLANDER [Esteva et al., 2001], MOISE<sup>+</sup> [Hübner et al., 2002], ...
- ▶ Specific anchors for situating organisations within the environment
  - e.g. embodied organisations [Piunti et al., 2009a]



## Organisation Oriented Programming (OOP)

Organisation as a **first class entity** in the multi-agent eco-system

- ▶ Clear distinction between description of the organisation **wrt** agents, **wrt** environment
- ▶ Different representations of the organisation:
  - ▶ **Organisation specification**
    - ▶ partially/totally accessible to the agents, to the environment, to the organisation
  - ▶ **Organisation entity**
    - ▶ Local representation in the mental state of the agents
      - ↪ possibly inconsistent with the other agents' representations
    - ▶ Global/local representation in the MAS
      - ↪ difficulty to manage and build such a representation in a distributed and decentralized setting
- ▶ Different sources of actions on (resp. of) the organisation by (resp. on) agents / environment / organisation



## Components of OOP: Organisation Management Infrastructure (OMI)

- ▶ *Coordination mechanisms*, i.e. support infrastructure
  - e.g. MADKIT [Gutknecht and Ferber, 2000], KARMA [Pynadath and Tambe, 2003], ...
- ▶ *Regulation mechanisms*, i.e. governance infrastructure
  - e.g. AMELI [Esteva et al., 2004], S-MOISE<sup>+</sup> [Hübner et al., 2006], ORA4MAS [Hübner et al., 2009], ...



## Components of OOP: Integration mechanisms

- ▶ *Agent* integration mechanisms  
allow agents to be aware of and to deliberate on:
  - ▶ entering/exiting the organisation
  - ▶ modification of the organisation
  - ▶ obedience/violation of norms
  - ▶ sanctioning/rewarding other agents
- e.g.  $\mathcal{J}$ -MOISE<sup>+</sup> [Hübner et al., 2007], Autonomy based reasoning [Carabelea, 2007], *ProsA*<sub>2</sub> Agent-based reasoning on norms [Ossowski, 1999], ...
- ▶ *Environment* integration mechanisms  
transform organisation into embodied organisation so that:
  - ▶ organisation may act on the environment (e.g. enact rules, regimentation)
  - ▶ environment may act on the organisation (e.g. count-as rules)
- e.g. [Piunti et al., 2009b], [Okuyama et al., 2008]



## Motivations for OOP: Constitutive point of view

- ▶ Organisation *helps* the agents to cooperate with the other agents by defining *common* cooperation schemes
  - ▶ global tasks
  - ▶ protocols
  - ▶ groups, responsibilities
- e.g. 'to bid' for a product on eBay is an *institutional action* only possible because eBay defines the rules for that very action
  - ▶ the bid protocol is a constraint but it also *creates* the action
- e.g. when a soccer team wants to play match, the organisation helps the members of the team to synchronise actions, to share information, etc



## Motivations for OOP: Applications point of view

- ▶ Current applications show an increase in
  - ▶ Number of agents
  - ▶ Duration and repetitiveness of agent activities
  - ▶ Heterogeneity of the agents, Number of designers of agents
  - ▶ Agent ability to act, to decide,
  - ▶ Action domains of agents, ...
  - ▶ Openness, scalability, dynamicity, ...
- ▶ More and more applications require the integration of human communities and technological communities (ubiquitous and pervasive computing), building connected communities (ICities) in which agents act on behalf of users
  - ▶ Trust, security, ..., flexibility, adaptation



## Motivations for OOP: Normative point of view

- ▶ MAS have two properties which seem contradictory:
  - ▶ a *global* purpose
  - ▶ *autonomous* agents
- ↪ While the autonomy of the agents is essential, it may cause loss in the global coherence of the system and achievement of the global purpose
- ▶ Embedding *norms* within the *organisation* of a MAS is a way to constrain the agents' behaviour towards the global purposes of the organisation, while explicitly addressing the autonomy of the agents within the organisation
  - ↪ Normative organisation
- e.g. when an agent adopts a role, it adopts a set of behavioural constraints that support the global purpose of the organisation. It may decide to obey or disobey these constraints



## Motivations for OOP: Agents point of view

Explicit representations, working environments are required so that the agents are able to reason about the organisation:

- ▶ to decide to enter into/leave from the organisation during execution
  - ↪ Organisation is no more closed
- ▶ to change/adapt the current organisation
  - ↪ Organisation is no more static
- ▶ to obey/disobey the organisation
  - ↪ Organisation is no more a regimentation



### 1 Origins and Fundamentals

### 2 Some OOP approaches

- AGR
- STEAM
- ISLANDER
- MOISE Framework

### 3 MOISE Organisation Modeling Language (OML)

### 4 MOISE Organisation Management Infrastructure (OMI)

### 5 MOISE Org. Embodiment Mechanisms for Cartago (E-O)

### 6 MOISE Org. Awareness Mechanisms in Jason (A-O)

### 7 Summary

## Motivations for OOP: Organisation point of view

Explicit representations, working environments are required so that the organisation is able to “reason” about itself and about the agents in order to ensure the achievement of its global purpose:

- ▶ to decide to let agents enter into/leave from the organisation during execution
  - ↪ Organisation is no more closed
- ▶ to decide to let agents change/adapt the current organisation
  - ↪ Organisation is no more static and blind
- ▶ to govern agents behaviour in the organisation (i.e. monitor, enforce, regiment)
  - ↪ Organisation is no more a regimentation

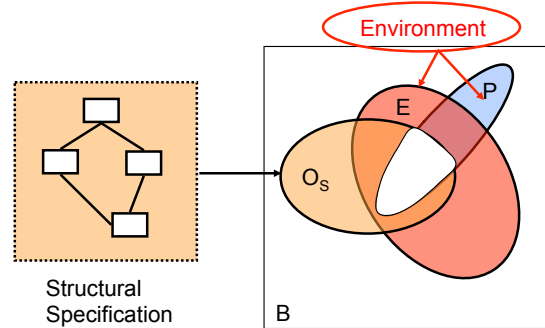


## AGR [Ferber and Gutknecht, 1998]

- ▶ Agent Group Role, previously known as AALAADIN
  - ▶ Agent: Active entity that plays roles within groups. An agent may have several roles and may belong to several groups.
  - ▶ Group: set of agents sharing common characteristics, i.e. context for a set of activities. Two agents can't communicate with each other if they don't belong to the same group.
  - ▶ Role: Abstract representation of the status, position, function of an agent within a group.
- ▶ OMI: the Madkit platform



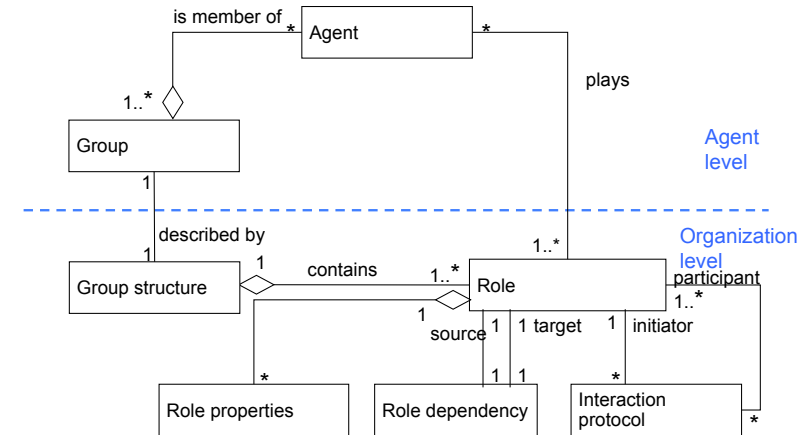
## AGR OML Modelling Dimensions



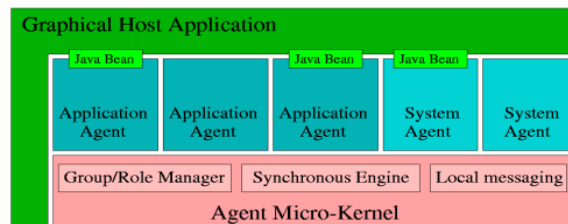
B: agents' possible behaviors  
P: agents' behaviors that lead to global purpose  
E: agents' possible behaviors constrained by the environment  
O<sub>s</sub>: agents' possible behaviors structurally constrained by the organization



## AGR OML

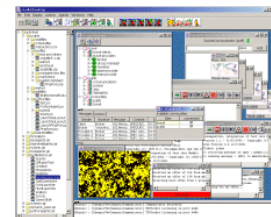


## AGR OMI: Madkit



Multi-Agent Development Kit

[www.madkit.org](http://www.madkit.org)

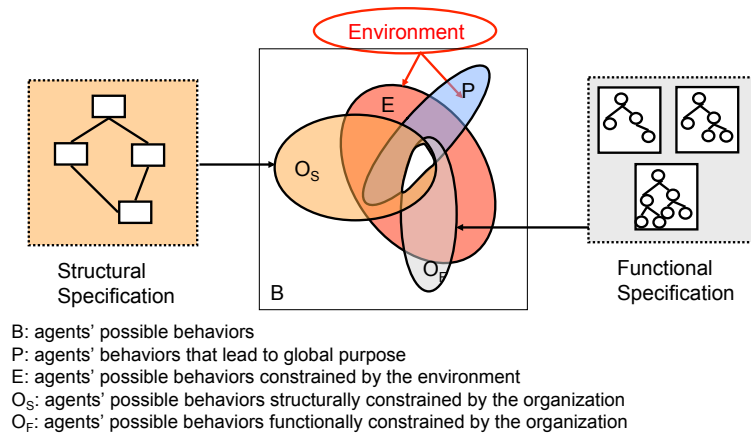


## STEAM [Tambe, 1997]

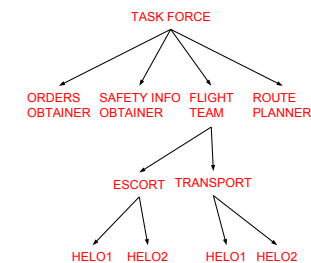
- ▶ Shell for TEAMwork is a general framework to enable agents to participate in teamwork.
  - ▶ Different applications: Attack, Transport, Robocup soccer
  - ▶ Based on an enhanced SOAR architecture and 300 domain independent SOAR rules
- ▶ Principles:
  - ▶ Team synchronization: Establish joint intentions, Monitor team progress and repair, Individual may fail or succeed in own role
  - ▶ Reorganise if there is a critical role failure
  - ▶ Reassign critical roles based on joint intentions
  - ▶ Decision theoretic communication
- ▶ Supported by the TEAMCORE OMI.



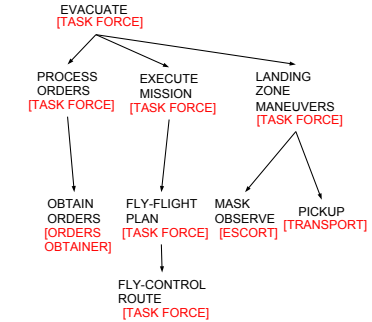
## STEAM OML Modelling Dimensions



## STEAM OML [Tambe, 1997]



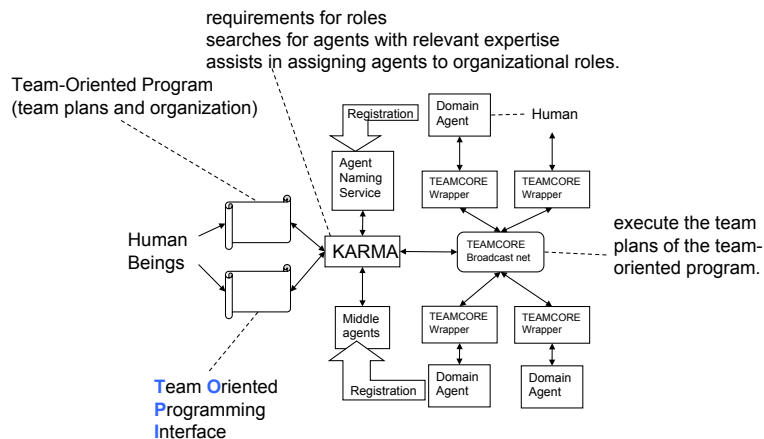
**Organization:** hierarchy of roles that may be filled by agents or groups of agents.



**Team Plan:**

- initial conditions,
- term. cond. : achievability, irrelevance, unachievability
- team-level actions.

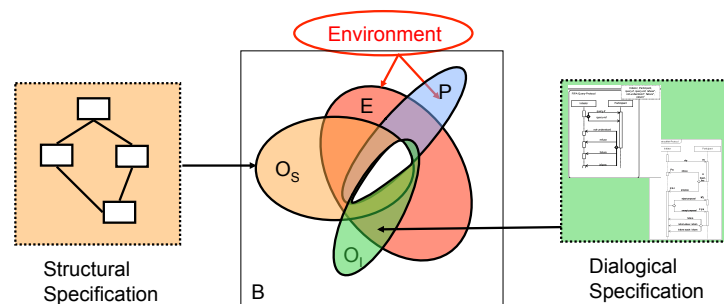
## STEAM OMI: TEAMCORE [Pynadath and Tambe, 2003]



## ISLANDER

- based on different influences: economics, norms, dialogues, coordination
- ~ electronic institutions
- combining different alternative views: dialogical, normative, coordination
- Institution Description Language: Performative structure (Network of protocols), Scene (multi-agent protocol), Roles, Norms
- AMELI as OMI

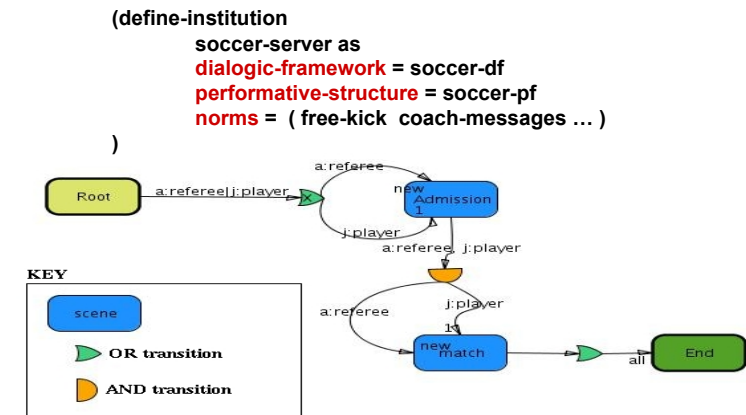
## ISLANDER OML Modelling Dimensions



B: agents' possible behaviors  
 P: agents' behaviors that lead to global purpose  
 E: agents' possible behaviors constrained by the environment  
 $O_s$ : agents' possible/permitted/obliged behaviors structurally constrained by the organisation  
 $O_i$ : agents' possible/permitted/obliged behaviors interactionally constrained by the organisation



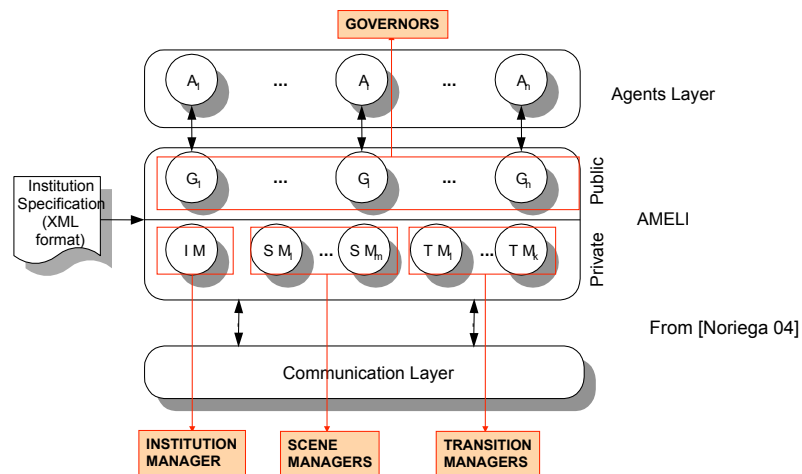
## ISLANDER OML: IDL [Esteva et al., 2001]



Performative Structure



## ISLANDER OMI: AMELI [Esteva et al., 2004]



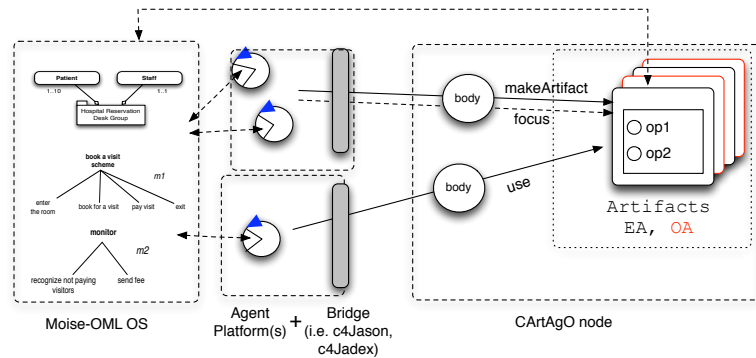
## $\mathcal{M}$ oise Framework

- $\mathcal{M}$ oise Organisation Modelling Language as Tag-based language (issued from  $\mathcal{M}$ oise [Hannoun et al., 2000],  $\mathcal{M}$ oise<sup>+</sup> [Hübner et al., 2002],  $\mathcal{M}$ oiseINST [Gâteau et al., 2005])
- OMI developed as an artefact-based working environment (ORA4MAS [Hübner et al., 2009] based on CARtAgO nodes) (refactoring of  $\mathcal{S}$ - $\mathcal{M}$ oise<sup>+</sup> [Hübner et al., 2006] and  $\mathcal{S}$ YNAI [Gâteau et al., 2005])
  - dedicated *organisational artefacts* that provide general services for the agents to work within an organisation
  - *organisational agents* that monitor and manage the functioning of the organisation
- Dedicated integration bridges for
  - Agents and Environment (c4Jason, c4Jadex [Ricci et al., 2009])
  - Environment and Organisation ([Piunti et al., 2009a])
  - Agents and Organisation ( $\mathcal{J}$ - $\mathcal{M}$ oise<sup>+</sup> [Hübner et al., 2007]  $\mathcal{M}$ oise<sup>+</sup> organisational events and actions integration within Jason)

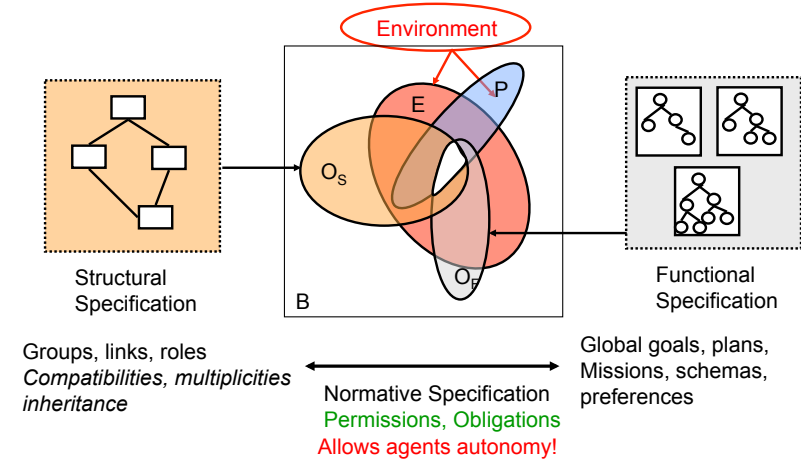




## Moise Framework as a Concrete Picture of OOP



## Moise Modelling Dimensions



### 1 Origins and Fundamentals

### 2 Some OOP approaches

### 3 MOISE Organisation Modeling Language (OML)

- Structural specification
- Functional specification
- Normative specification

### 4 MOISE Organisation Management Infrastructure (OMI)

### 5 MOISE Org. Embodiement Mechanisms for Cartago (E-O)

### 6 MOISE Org. Awareness Mechanisms in Jason (A-O)

### 7 Summary

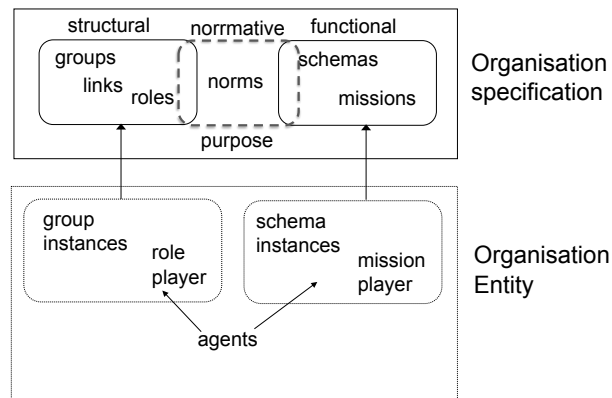


## Moise OML

- ▶ OML for defining organisation specification and organisation entity
- ▶ Three independent dimensions [Hübner et al., 2007] (↪ well adapted for the reorganisation concerns):
  - ▶ *Structural*: Roles, Groups
  - ▶ *Functional*: Goals, Missions, Schemes
  - ▶ *Normative*: Norms (obligations, permissions, interdictions)
- ▶ Abstract description of the organisation for
  - ▶ the designers
  - ▶ the agents
    - ↪  $\mathcal{J}$ -MOISE [Hübner et al., 2007]
  - ▶ the Organisation Management Infrastructure
    - ↪ ORA4MAS [Hübner et al., 2009]



## Moise OML global picture



## Structural Specification

- Specifies the structure of an MAS along three levels:
  - Individual* with *Role*
  - Social* with *Link*
  - Collective* with *Group*
- Components:
  - Role*: label used to assign constraints on the behavior of agents playing it
  - Link*: relation between roles that directly constrains the agents in their interaction with the other agents playing the corresponding roles
  - Group*: set of links, roles, compatibility relations used to define a shared context for agents playing roles in it



## Structural specification

- Defined with the tag **structural-specification** in the context of an **organisational-specification**
- One section for definition of all the roles participating to the structure of the organisation (**role-definitions** tag)
- Specification of the group including all sub-group specifications (**group-specification** tag)

### Example

```
<organisational-specification>
  <structural-specification>
    <role-definitions> ... </role-definitions>
    <group-specification id="xxx">
      ...
    </group-specification>
  </structural-specification>
  ...
</organisational-specification>
```



## Role specification

- Role definition(**role** tag) in **role-definitions** section, is composed of:
  - identifier of the role (**id** attribute of **role** tag)
  - inherited roles (**extends** tag) - by default, all roles inherit of the **soc** role -

### Example

```
<role-definitions>
  <role id="player" />
  <role id="coach" />
  <role id="middle"> <extends role="player"/> </role>
  <role id="leader"> <extends role="player"/> </role>
  <role id="r1">
    <extends role="r2" />
    <extends role="r3" />
  </role>
  ...
</role-definitions>
```



## Group specification

- ▶ Group definition (**group-specification** tag) is composed of:
  - ▶ group identifier (**id** attribute of **group-specification** tag)
  - ▶ roles participating to this group and their cardinality (**roles** tag and **id**, **min**, **max**), i.e. min. and max. number of agents that should adopt the role in the group (default is 0 and unlimited)
  - ▶ links between roles of the group (**link** tag)
  - ▶ subgroups and their cardinality (**sub-groups** tag)
  - ▶ formation constraints on the components of the group (**formation-constraints**)

### Example

```
<group-specification id="team">
  <roles>
    <role id="coach" min="1" max="2"/> ...
  </roles>
  <links> ... </links>
  <sub-groups> ... </sub-groups>
  <formation-constraints> ... </formation-constraints>
</group-specification>
```

WI Master, Nov 2011

41 / 107

## Link specification

- ▶ Link definition (**link** tag) included in the group definition is composed of:
  - ▶ role identifiers (**from**, **to**)
  - ▶ type (**type**) with one of the following values: **authority**, **communication**, **acquaintance**
  - ▶ scope of the link (**scope**): **inter-group**, **intra-group**
  - ▶ validity in sub-groups: if **extends-sub-group** set to true, the link is also valid in all sub-groups (default false)

### Example

```
<link from="coach"
      to="player"
      type="authority"
      scope="inter-group"
      extends-sub-groups="true" />
```

WI Master, Nov 2011

42 / 107

## Formation constraint specification

- ▶ Formation constraints definition (**formation-constraints** tag) in a group definition is composed of:
  - ▶ compatibility constraints (**compatibility** tag) between roles (**from**, **to**), with a **scope**, **extends-sub-groups** and directions (**bi-dir**)

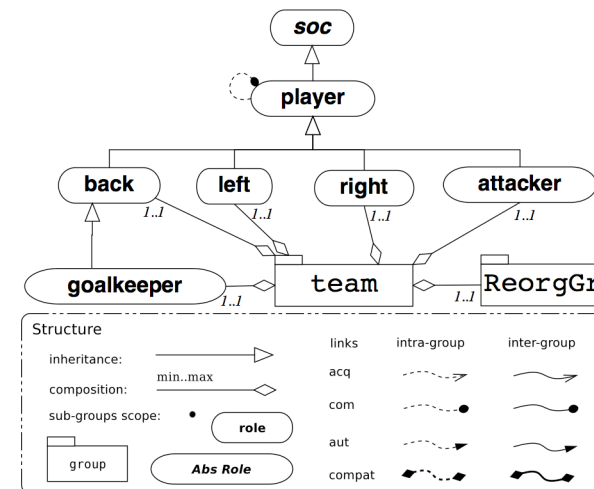
### Example

```
<formation-constraints>
  <compatibility from="middle"
                to="leader"
                scope="intra-group"
                extends-sub-groups="false"
                bi-dir="true"/>
  ...
</formation-constraints>
```

WI Master, Nov 2011

43 / 107

## Structural specification example (1)

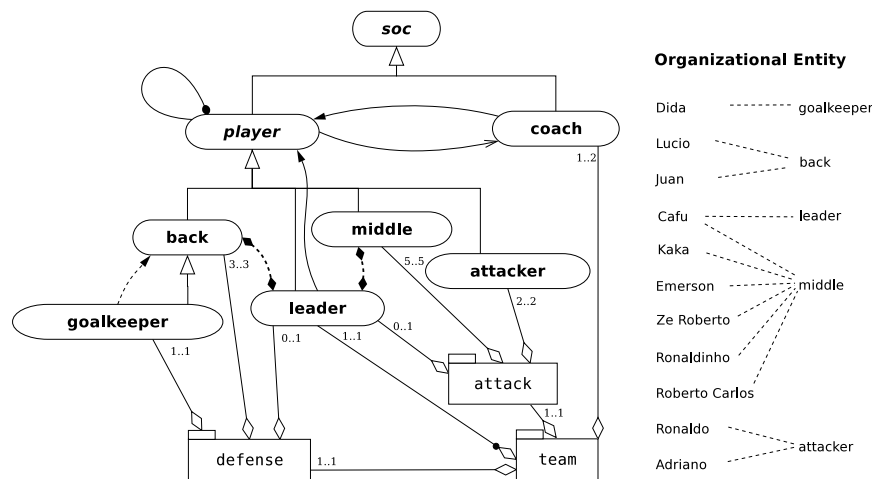


Graphical representation of structural specification of Joj Team

WI Master, Nov 2011

44 / 107

## Structural specification example (2)



Graphical representation of structural specification of 3-5-2 Joj Team

WI Master, Nov 2011

45 / 107

## Functional Specification

- Specifies the expected behaviour of an MAS in terms of *goals* along two levels:
  - *Collective* with *Scheme*
  - *Individual* with *Mission*
- Components:
  - *Goals*:
    - *Achievement goal* (default type). Goals of this type should be declared as satisfied by the agents committed to them, when achieved
    - *Maintenance goal*. Goals of this type are not satisfied at a precise moment but are pursued while the scheme is running. The agents committed to them do not need to declare that they are satisfied
  - *Scheme*: global goal decomposition tree assigned to a group
    - Any scheme has a root goal that is decomposed into subgoals
  - *Missions*: set of coherent goals assigned to roles within norms

WI Master, Nov 2011

46 / 107

## Functional specification

- Defined with the tag `functional-specification` in the context of an `organisational-specification`
- Specification in sequence of the different schemes participating to the expected behaviour of the organisation

### Example

```
<functional-specification>
  <scheme id="sideAttack" >
    <goal id="dogoal" > ... </goal>
    <mission id="m1" min="1" max="5">
      ...
    </mission>
    ...
  </scheme>
  ...
</functional-specification>
```

WI Master, Nov 2011

47 / 107

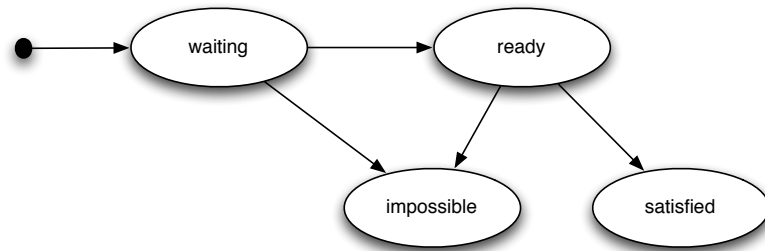
## Scheme specification

- Scheme definition (`scheme` tag) is composed of:
  - identifier of the scheme (`id` attribute of `scheme` tag)
  - the root goal of the scheme with the plan aiming at achieving it (`goal` tag)
  - the set of missions structuring the scheme (`mission` tag)
- Goal definition within a scheme (`goal` tag) is composed of:
  - an identifier (`id` attribute of `goal` tag)
  - a `type` (`achievement` default or `maintenance`)
  - min. number of agents that must satisfy it (`min`) (default is "all")
  - optionally, an argument (`argument` tag) that must be assigned to a value when the scheme is created
  - optionally a plan
- Plan definition attached to a goal (`plan` tag) is composed of
  - one and only one operator (`operator` attribute of `plan` tag) with `sequence`, `choice`, `parallel` as possible values
  - set of goal definitions (`goal` tag) concerned by the operator

WI Master, Nov 2011

48 / 107

## Goal States



**waiting** initial state

**ready** goal pre-conditions are satisfied & scheme is well-formed

**satisfied** agents committed to the goal have achieved it

**impossible** the goal is impossible to be satisfied



## Scheme specification example

```

<scheme id="sideAttack">
  <goal id="scoreGoal" min="1" >
    <plan operator="sequence">
      <goal id="g1" min="1" ds="get the ball" />
      <goal id="g2" min="3" ds="to be well placed">
        <plan operator="parallel">
          <goal id="g7" min="1" ds="go toward the opponent's field" />
          <goal id="g8" min="1" ds="be placed in the middle field" />
          <goal id="g9" min="1" ds="be placed in the opponent's goal area" />
        </plan>
      </goal>
    </plan>
  </goal>
  <goal id="g3" min="1" ds="kick the ball to the m2Ag" >
    <argument id="M2Ag" />
  </goal>
  <goal id="g4" min="1" ds="go to the opponent's back line" />
  <goal id="g5" min="1" ds="kick the ball to the goal area" />
  <goal id="g6" min="1" ds="shot at the opponent's goal" />
</plan>
</goal>
  ...

```



## Mission specification

- Mission definition (**mission** tag) in the context of a scheme definition, is composed of:
  - identifier of the mission (**id** attribute of **mission** tag)
  - cardinality of the mission **min** (0 is default), **max** (unlimited is default) specifying the number of agents that can be committed to the mission
  - the set of goal identifiers (**goal** tag) that belong to the mission

### Example

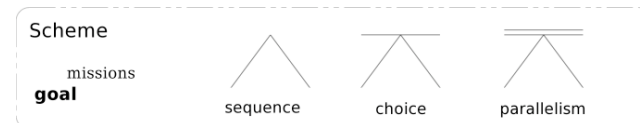
```

<scheme id="sideAttack">
  ... the goals ...
  <mission id="m1" min="1" max="1">
    <goal id="scoreGoal" /> <goal id="g1" />
    <goal id="g3" /> ...
  </mission>
  ...
</scheme>

```



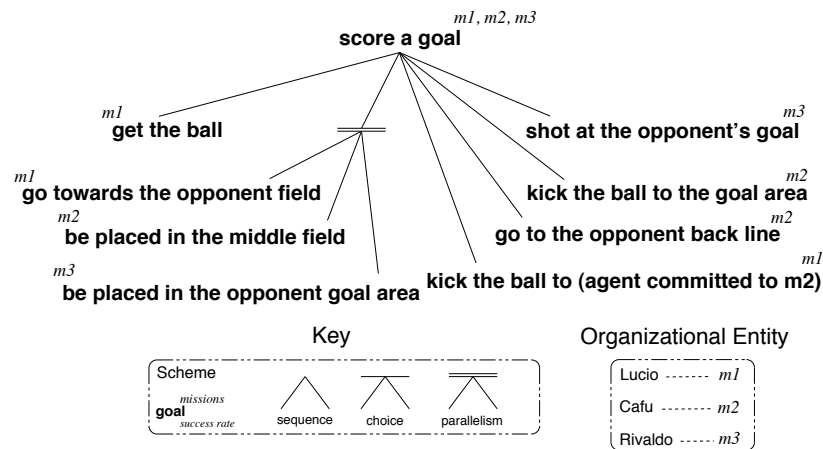
## Functional specification example (1)



Graphical representation of social scheme for joj team



## Functional specification example (2)



Graphical representation of social scheme "side\_attack" for joj team

## Normative Specification

- ▶ Explicit relation between the functional and structural specifications
- ▶ Permissions and obligations to commit to missions in the context of a role
- ▶ Makes explicit the normative dimension of a role

## Normative specification

- ▶ Defined with the tag **normative-specification** in the context of an **organisational-specification**
- ▶ Specification in sequence of the different norms participating to the governance of the organisation

### Example

```
<normative-specification>
  <norm id="n1" ... />
  ...
  <norm id="..." ... />
</normative-specification>
```

## Norm specification

- ▶ Norm definition (**norm** tag) in the context of a **normative-specification** definition, is composed of:
  - ▶ the identifier of the norm (**id**)
  - ▶ the type of the norm (**type**) with **obligation**, **permission** as possible values
  - ▶ optionally a condition of activation (**condition**) with the following possible expressions:
    - ▶ checking of properties of the organisation (e.g. **#role\_compatibility**, **#mission\_cardinality**, **#role\_cardinality**, **#goal\_non\_compliance**)
    - ↪ unregimentation of organisation properties !!!
    - ▶ (un)fulfillment of an obligation stated in a particular norm (**unfulfilled**, **fulfilled**)
  - ▶ the identifier of the role (**role**) on which the role is applied
  - ▶ the identifier of the mission (**mission**) concerned by the norm
  - ▶ optionally a time constraint (**time-constraint**)

## Norm Specification – example

role	deontic	mission		TTF
back	obliged	m1	get the ball, go ...	1 minute
left	obliged	m2	be placed at ..., kick ...	3 minute
right	obliged	m2		1 day
attacker	obliged	m3	kick to the goal, ...	30 seconds

```

<norm id = "n1" type="obligation"
  role="back" mission="m1" time-constraint="1 minute"/>
...
<norm id = "n4" type="obligation"
  condition="unfulfilled(obligation(_,n2,_,_))"
  role="coach" mission="ms" time-constraint="3 hour"/>
...

```



- 1 Origins and Fundamentals
- 2 Some OOP approaches
- 3 .MOISE Organisation Modeling Language (OML)
- 4 .MOISE Organisation Management Infrastructure (OMI)
- 5 .MOISE Org. Embodiement Mechanisms for Cartago (E-O)
- 6 .MOISE Org. Awareness Mechanisms in Jason (A-O)
- 7 Summary

## Organisation Entity Dynamics

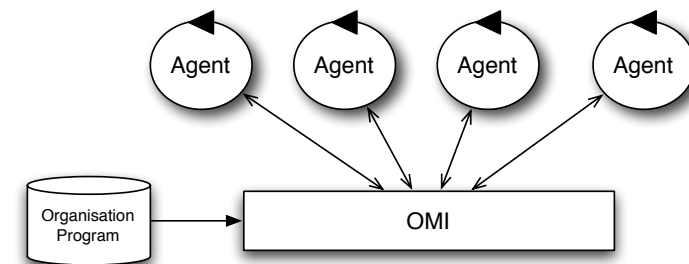
- 1 Organisation is created (by the agents)
  - ▶ instances of groups
  - ▶ instances of schemes
- 2 Agents enter into groups *adopting* roles
- 3 Groups become *responsible* for schemes
  - ▶ Agents from the group are then obliged to commit to missions in the scheme
- 4 Agents *commit* to missions
- 5 Agents *fulfil* mission's goals
- 6 Agents leave schemes and groups
- 7 Schemes and groups instances are destroyed



## Organisation management infrastructure (OMI)

### Responsibility

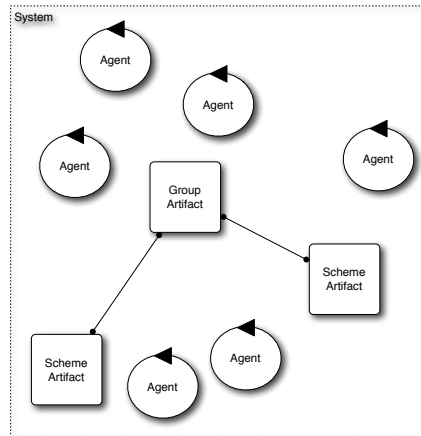
- ▶ Managing – coordination, regulation – the agents' execution within organisation defined by an organisational specification



(e.g. MadKit, AMELI,  $\mathcal{S}$ -MOISE<sup>+</sup>, ...)



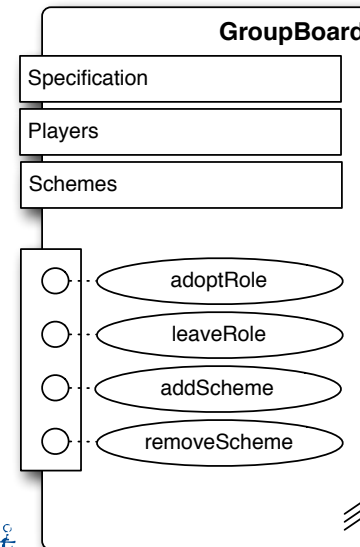
## ORA4MAS



- ▶ Based on A&A and *MOISE*
- ▶ Agents' working environment is instrumented with Organisational Artifacts (OA) offering "organisational" actions
- ▶ Agents create, handle, perceive and act on OAs
- ▶ OAs are in charge of *regimentations*, detection and evaluation of norms compliance
- ▶ Agents are in charge of decisions about sanctions
- ▶ *Distributed* management of the organisation



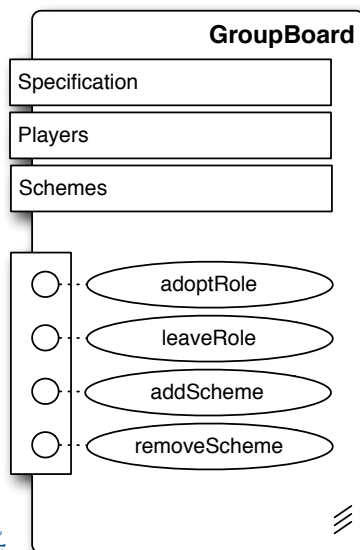
## ORA4MAS – GroupBoard artifact

*Operations:*

- ▶ **adoptRole(role)**: the agent executing this operation tries to adopt a **role** in the group
- ▶ **leaveRole(role)**
- ▶ **addScheme(schid)**: the group starts to be responsible for the scheme managed by the SchemeBoard **schId**
- ▶ **removeScheme(schid)**



## ORA4MAS – GroupBoard artifact

*Observable Properties:*

- ▶ **specification**: the specification of the group in the OS (an object of class *moise.os.ss.Group*)
- ▶ **players**: a list of agents playing roles in the group. Each element of the list is a pair (agent x role)
- ▶ **schemes**: a list of scheme identifiers that the group is responsible for



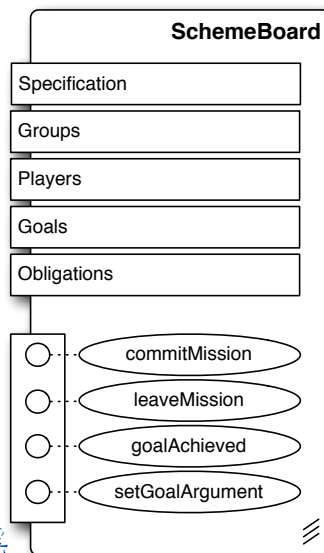
## ORA4MAS – GroupBoard artifact

- ▶ Signals (parameter *o* has the following form "obligation(to whom, reason, what, deadline)"):
  - ▶ **obl\_created(o)**: the obligation *o* is created
  - ▶ **obl\_fulfilled(o)**: the obligation *o* is fulfilled
  - ▶ **obl\_unfulfilled(o)**: the obligation *o* is unfulfilled (e.g. by timeout)
  - ▶ **obl\_inactive(o)**: the obligation *o* is inactive (e.g. its condition does not hold anymore)
  - ▶ **norm\_failure(f)**: the failure *f* has happened (e.g. due some regimentation violation)





## ORA4MAS – SchemeBoard artifact



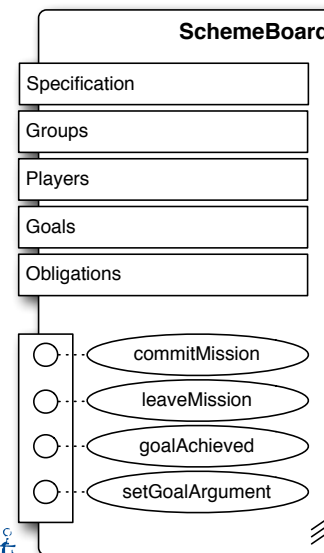
### Operations:

- ▶ **commitMission(mission)** and **leaveMission**: operations to “enter” and “leave” the scheme
- ▶ **goalAchieved(goal)**: defines that some goal is achieved by the agent performing the operation
- ▶ **setGoalArgument(goal, argument, value)**: defines the value of some goal’s argument

WI Master, Nov 2011

65 / 107

## ORA4MAS – SchemeBoard artifact



### Observable Properties:

- ▶ **specification**: the specification of the scheme in the OS
- ▶ **groups**: a list of groups responsible for the scheme
- ▶ **players**: a list of agents committed to the scheme. Each element of the list is a pair (agent, mission)
- ▶ **goals**: a list with the current state of the goals
- ▶ **obligations**: list of obligations currently active in the scheme

WI Master, Nov 2011

66 / 107

## ORA4MAS – SchemeBoard artifact

- ▶ Signals (parameter o is of the form: **obligation(to whom, reason, what, deadline)**):
  - ▶ **obl\_created(o)**: the obligation o is created
  - ▶ **obl\_fulfilled(o)**: the obligation o is fulfilled
  - ▶ **obl\_unfulfilled(o)**: the obligation o is unfulfilled (e.g. by timeout)
  - ▶ **obl\_inactive(o)**: the obligation o is inactive (e.g. its condition does not hold anymore)
  - ▶ **norm\_failure(f)**: the failure f has happened (e.g. due some regimentation violation)

WI Master, Nov 2011

67 / 107

## Organisational Artifact Implementation

- ▶ Organisational artifacts are programmed with a Normative Programming Language (NPL) [Hübner et al., 2010]
- ▶ The NPL *norms* have
  - ▶ an activation condition
  - ▶ a consequence
- ▶ two kinds of consequences are considered
  - ▶ regimentations
  - ▶ obligations

### Example (norm)

```
norm n1: plays(A,writer,G) -> fail.
```

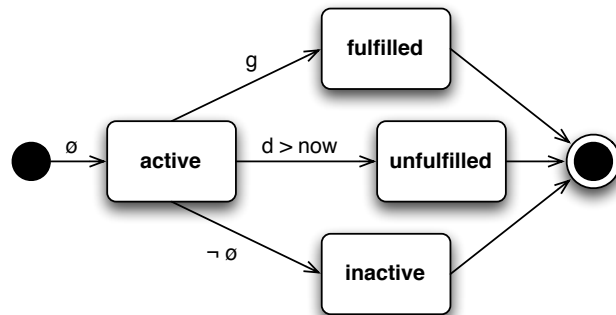
or

```
norm n1: plays(A,writer,G)
-> obligation(A,n1,plays(A,editor,G),
'now + 3 min').
```

WI Master, Nov 2011

68 / 107

## Obligations life cycle



- ▶  $\phi$ : activation condition (e.g. play a role)
- ▶  $g$ : the obligation (e.g. commit to a mission)

## OS in *Moise* OML to NOPL translation

### Example (role cardinality norm – regimentation)

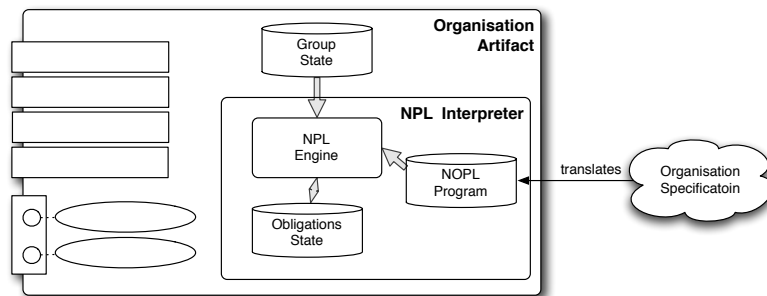
```
group_role(writer,1,5).
```

```
norm ncar: group_role(R,_,M) &
    rplayers(R,G,V) & V > M
-> fail(role_cardinality(R,G,V,M)).
```

### Example (role cardinality norm – agent decision)

```
norm ncar: group_role(R,_,M) &
    rplayers(R,G,V) & V > M &
    plays(E,editor,G)
-> obligation(E,ncar,committed(E,ms,_),
    'now + 1 hour').
```

## Organisational Artifact Architecture



Signals ( $o$  = obligation(to whom, reason, what, deadline)):

- ▶ `obl_created( $o$ )`: the obligation  $o$  is created
- ▶ `obl_fulfilled( $o$ )`: the obligation  $o$  is fulfilled
- ▶ `obl_unfulfilled( $o$ )`: the obligation  $o$  is unfulfilled
- ▶ `obl_inactive( $o$ )`: the obligation  $o$  is inactive
- ▶ `norm_failure( $f$ )`: the failure  $f$  has happened

### 1 Origins and Fundamentals

### 2 Some OOP approaches

### 3 *Moise* Organisation Modeling Language (OML)

### 4 *Moise* Organisation Management Infrastructure (OMI)

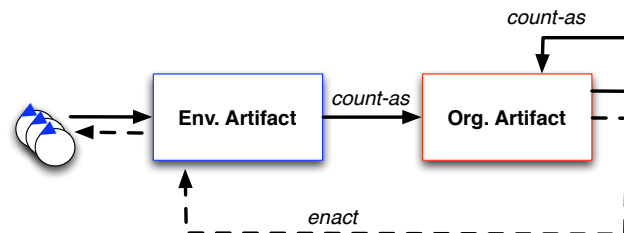
### 5 *Moise* Org. Embodiement Mechanisms for Cartago (E-O)

### 6 *Moise* Org. Awareness Mechanisms in Jason (A-O)

### 7 Summary

## Environment integration

- ▶ Organisational Artifacts enable organisation and environment integration
- ▶ Embodied organisation [Piunti et al., 2009a]



status: ongoing work

- 1 Origins and Fundamentals
- 2 Some OOP approaches
- 3 .MOISE Organisation Modeling Language (OML)
- 4 .MOISE Organisation Management Infrastructure (OMI)
- 5 .MOISE Org. Embodiement Mechanisms for Cartago (E-O)
- 6 .MOISE Org. Awareness Mechanisms in Jason (A-O)
  - Organisational actions
  - Organisational Perception
  - Organisational goals
  - Example

- 7 Summary

## Constitutive rules

### Count-As rule

An event occurring on an artifact, in a particular context, may count-as an institutional event

- ▶ transforms the events created in the working environment into activation of an organisational operation
- ↪ indirect automatic updating of the organisation

### Enact rule

An event produced on an organisational artifact, in a specific institutional context, may “enact” change and updating of the working environment (i.e., to promote equilibrium, avoid undesirable states)

- ▶ Installing automated control on the working environment
- ▶ Even without the intervention of organisational/staff agents (regimenting actions on physical artifacts, enforcing sanctions,

## Agent integration

- ▶ Agents can interact with organisational artifacts as with ordinary artifacts by perception and action
- ↪ Any Agent Programming Language integrated with CArtaGO can use organisational artifacts

Agent integration provides some “internal” tools for the agents to simplify their interaction with the organisation:

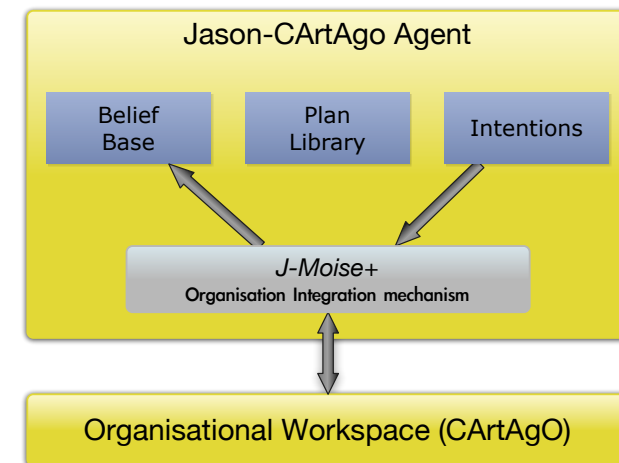
- ▶ maintenance of a local copy of the organisational state
- ▶ production of *organisational events*
- ▶ provision of *organisational actions*



## J-Moise: Jason + Moise

- ▶ Agents are programmed with *Jason*
- ↪ BDI agents (reactive planning) – suitable abstraction level
- ▶ The programmer has the possibility to express sophisticated recipes for adopting roles, committing to missions, fulfilling/violating norms, ...
- ▶ Organisational information is made accessible in the mental state of the agent as beliefs
- ▶ Integration is totally independent of the distribution/communication layer

## J-Moise: Jason + Moise– General view



## Organisational actions in Jason I

- ▶ For groups:
  - ▶ create\_group
  - ▶ remove\_group

### Example

```

...
.my_name(Me);
join_workspace(ora4mas,"",user_id(Me));
create_group(
    mypaper,      // group identification
    "wp-os.xml",  // specification file
    wpgroup,      // group type
    false,        // monitoring scheme
    true);        // GUI
adopt_role(editor,mypaper);
  
```

## Organisational actions in Jason II

- ▶ For schemes:
  - ▶ create\_scheme
  - ▶ add\_responsible\_group
  - ▶ remove\_scheme
  - ▶ goal\_achieved

### Example

```

create_scheme(
    s45,
    "wp-os.xml",
    writePaperSch,
    false,
    true);
add_responsible_group(s45,mypaper);
commit_mission(mManager, S).
  
```

## Organisational actions in Jason III

- ▶ For roles:
  - ▶ adopt\_role
  - ▶ remove\_role
- ▶ For missions:
  - ▶ commit\_mission
  - ▶ remove\_mission
- ▶ Those actions usually are executed under *regimentation* (to avoid an inconsistent organisational state) e.g. the adoption of role is constrained by
  - ▶ the cardinality of the role in the group
  - ▶ the compatibilities of the roles played by the agent



## Organisational perception

When an agent focus on an Organisational Artifact, the observable properties (Java objects) are translated to beliefs with the following predicates:

- ▶ group\_specification
- ▶ scheme\_specification
- ▶ plays(agent, role, group)
- ▶ responsible\_group(group, scheme)
- ▶ commitment(agent, mission, scheme)
- ▶ goal(scheme, goal, list of committed agents, list of agent that achieved the goal, state of the goal)
- ▶ obligation(agent,norm,goal,dead line)



## Organisational perception – example

```

:: Jason Mind Inspector ::
peption
commitment(agp3,mParticipant,round_1_12){artifact(round_1_12),source(ora4mas)}
current_winner(agp2){source(self)}
goal_state(round_1_12,round(1,12),[],[],satisfied){artifact(round_1_12),source(ora4mas)}
goal_state(round_1_12,finish(NewW),[bob],[bob],satisfied){artifact(round_1_12),source(ora4mas)}
goal_state(round_1_12,bid,[agp5,agp4,agp1,agp2,agp3],[agp2],satisfied){artifact(round_1_12),s
goal_state(round_1_12,start(null),[bob],[bob],satisfied){artifact(round_1_12),source(ora4mas)}
group_specification(auctionGroup,[role(auctioneer,1,1,[]),link(communication,participant,intra
participant,0,300,[]),link(communication,auctioneer,intra_group),link(communication,participant
[],properties([]))){artifact(auction),source(ora4mas)}
object(1,12){source(self)}
play(agp1,participant,auction){artifact(auction),source(ora4mas)}
play(agp2,participant,auction){artifact(auction),source(ora4mas)}

```



## Handling organisational events in Jason

Whenever something changes in the organisation, the agent architecture updates the agent belief base accordingly producing events (belief update from perception)

### Example (new agent entered the group)

```
+play(Ag, boss, GId) <- .send(Ag, tell, hello).
```

### Example (change in goal state)

```
+goal_state(Scheme, wsecs, __, __, satisfied)
: .my_name(Me) & commitment(Me, mCol, Scheme)
<- leave_mission(mColaborator, Scheme).
```

### Example (signals)

```
+norm_failure(N) <- .print("norm failure event: ", N).
```



## Typical plans for obligations

### Example

```
+obligation(Ag,Norm,committed(Ag,Mission,Scheme),DeadLine)
: .my_name(Ag)
<- .print("I am obliged to commit to ",Mission);
   commit_mission(Mission,Scheme).

+obligation(Ag,Norm,achieved(Sch,Goal,Ag),DeadLine)
: .my_name(Ag)
<- .print("I am obliged to achieve goal ",Goal);
   !Goal[scheme(Sch)];
   goal_achieved(Goal,Sch).

+obligation(Ag,Norm,What,DeadLine)
: .my_name(Ag)
<- .print("I am obliged to ",What,
          ", but I don't know what to do!").
```



## Writing paper example

### Organisation Specification

```
<organisational-specification
  <structural-specification>
    <role-definitions>
      <role id="author" />
      <role id="writer"> <extends role="author"/> </role>
      <role id="editor"> <extends role="author"/> </role>
    </role-definitions>

    <group-specification id="wpgroup">
      <roles>
        <role id="writer" min="1" max="5" />
        <role id="editor" min="1" max="1" />
      </roles>
      ...
```



## Writing paper sample I

### Execution

```
jaime action: jmoise.create_group(wpgroup)
all perception: group(wpgroup,g1)[owner(jaime)]
jaime action: jmoise.adopt_role(editor,g1)
olivier action: jmoise.adopt_role(writer,g1)
jomi action: jmoise.adopt_role(writer,g1)
all perception:
  play(jaime,editor,g1)
  play(olivier,writer,g1)
  play(jomi,writer,g1)
```



## Writing paper sample II

### Execution

```
jaime action: jmoise.create_scheme(writePaperSch, [g1])
all perception: scheme(writePaperSch,s1)[owner(jaime)]
all perception: scheme_group(s1,g1)
jaime perception:
  permission(s1,mManager)[role(editor),group(wpgroup)]
jaime action: jmoise.commit_mission(mManager,s1)
olivier perception:
  obligation(s1,mColaborator)[role(writer),group(wpgroup),
  obligation(s1,mBib)[role(writer),group(wpgroup)]
olivier action: jmoise.commit_mission(mColaborator,s1)
olivier action: jmoise.commit_mission(mBib,s1)
```



## Writing paper sample III

### Execution

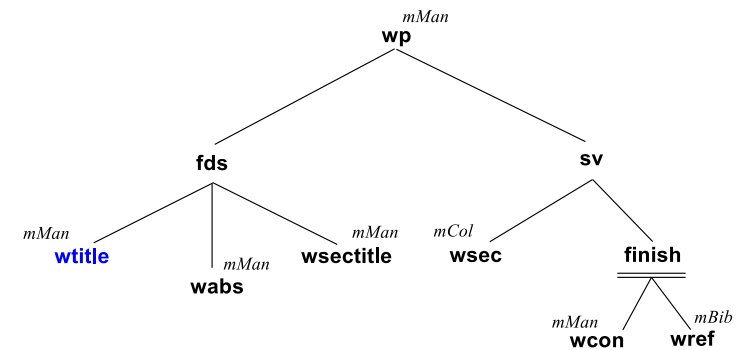
jomi perception:  
 obligation(s1,mColaborator)[role(writer),group(wpgroup),  
 obligation(s1,mBib)[role(writer),group(wpgroup)]

jomi action: jmoise.commit\_mission(mColaborator,s1)

all perception:  
 commitment(jaime,mManager,s1)  
 commitment(olivier,mColaborator,s1)  
 commitment(olivier,mBib,s1)  
 commitment(jomi,mColaborator,s1)

## Writing paper sample IV

### Execution



all perception: goal.state(s1,\*,unsatisfied)



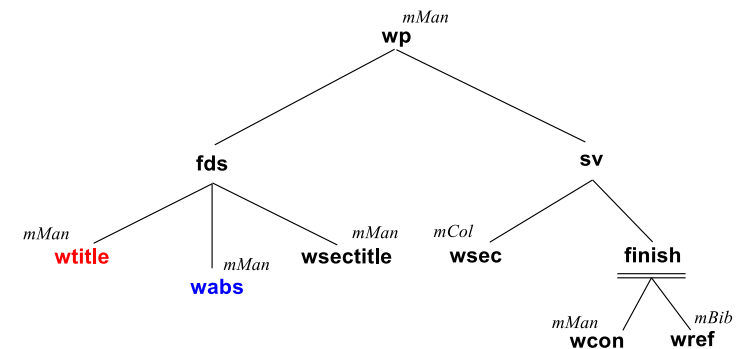
## Writing paper sample V

### Execution

jaime (only wtitle is possible, Jaime should work)  
 event: +!wtitle  
 action: jmoise.set\_goal\_state(s1,wtitle,satisfied)

## Writing paper sample VI

### Execution

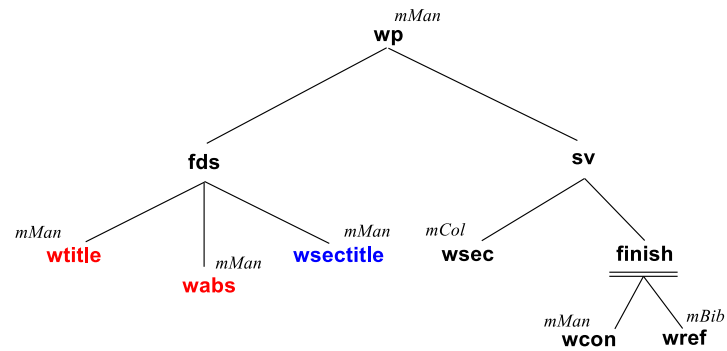


jaime event: +!wabs  
 action: jmoise.set\_goal\_state(s1,wabs,satisfied)



## Writing paper sample VII

Execution

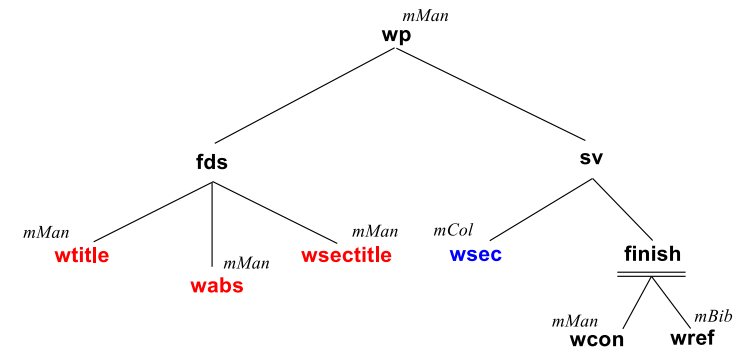


jaime event:  $+!wsectitles$   
 action: jmoise.set\_goal\_state(s1,wsectitles,satisfied)



## Writing paper sample VIII

Execution

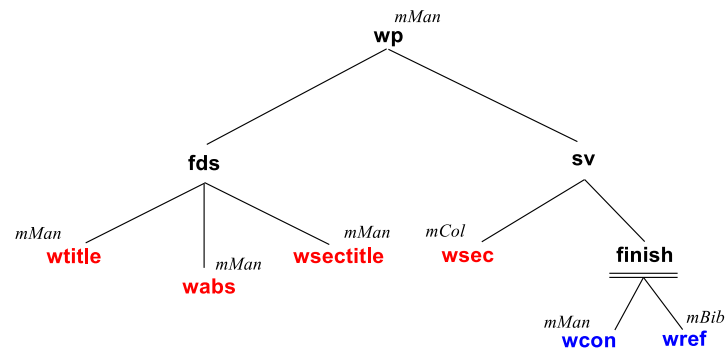


olivier, jomi event:  $+!wsecs$   
 action: jmoise.set\_goal\_state(s1,wsecs,satisfied)



## Writing paper sample IX

Execution



jaime event:  $+!wcon$ ; ...  
 olivier event:  $+!wref$ ; ...



## Writing paper sample X

Execution

all action: jmoise.remove\_mission(s1)  
 jaime action: jmoise.jmoise.remove\_scheme(s1)





## Useful tools — Mind inspector

```

play(gaucha0,herder,gr_herding_grp_13){source(orgManager)}.
play(gaucha4,herdboy,gr_herding_grp_13){source(orgManager)}.
play(gaucha5,herdboy,gr_herding_grp_13){source(orgManager)}.
pos(45,44,128){source(percept)}.
scheme(herd_sch,sch_herd_sch_18){owner(gaucha3),source(orgManager)}.
scheme(herd_sch,sch_herd_sch_12){owner(gaucha1),source(orgManager)}.
scheme_group(sch_herd_sch_12,gr_herding_grp_13){source(orgManager)}.
steps(700){source(self)}.
target(6,44){source(gaucha1)}.

```

- Rules	random_pos(X,Y):- (pos(AgX,AgY,_418) & (jia.random(RX,40) & ((RX > 5) & ((X = ((RX-20)+AgX)) & ((X >		
- Intentions	Sel Id	Pen	Intended Means Stack (hide details)
	16927	suspended-self	+lbe_in_formation[scheme(sch_herd_sch_12),mission(hel +lbe_in_formation[scheme(Sch),mission(Mission)]



## Summary

- ▶ *Jason*
  - ▶ declarative and goal oriented programming
  - ▶ goal patterns (maintenance goal)
  - ▶ meta-programming (.drop intention( [group(g1)])
  - ▶ customisations (integration with the simulator and the organisation)
  - ▶ internal actions (code in Java)
  - ↪ good programming style
- ▶ *MOISE* Framework
  - ▶ definition of groups and roles
  - ▶ allocation of goals to agents based on their roles
  - ▶ to change the team, we (developers) simply change the organisation
  - ▶ global orchestration
  - ↪ team strategy defined at a high level



## Summary

- ▶ Ensures that the agents follow some of the constraints specified for the organisation
- ▶ Helps the agents to work together
- ▶ The organisation is *interpreted at runtime*, it is not hardwired in the agents code
- ▶ The agents 'handle' the organisation (i.e. their artifacts)
- ▶ It is suitable for open systems as no specific agent architecture is required

- ▶ All available as open source at

<http://moise.souceforge.net>






## Bibliography I

- Bernoux, P. (1985).  
*La sociologie des organisations*.  
Seuil, 3ème edition.
- Boella, G., Torre, L., and Verhagen, H. (2008).  
Introduction to the special issue on normative multiagent systems.  
*Autonomous Agents and Multi-Agent Systems*, 17(1):1–10.
- Carabelea, C. (2007).  
*Reasoning about autonomy in open multi-agent systems - an approach based on the social power theory*.  
in french, ENS Mines Saint-Etienne.
- Esteva, M., Rodriguez-Aguar, J. A., Sierra, C., Garcia, P., and Arcos, J. L. (2001).  
On the formal specification of electronic institutions.  
In Dignum, F. and Sierra, C., editors, *Proceedings of the Agent-mediated Electronic Commerce*, LNAI 1191, pages 126–147, Berlin. Springer.





## Bibliography II

-  Esteva, M., Rodríguez-Aguilar, J. A., Rosell, B., and Arcos, J. L. (2004). AMELI: An agent-based middleware for electronic institutions. In Jennings, N. R., Sierra, C., Sonenberg, L., and Tambe, M., editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2004)*, pages 236–243, New York. ACM.
-  Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agents systems. In Demazeau, Y., editor, *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)*, pages 128–135. IEEE Press.
-  Gasser, L. (2001). Organizations in multi-agent systems. In *Pre-Proceeding of the 10th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'2001)*, Annecy.






## Bibliography IV

-  Hübner, J. F., Boissier, O., and Bordini, R. H. (2010). A normative organisation programming language for organisation management infrastructures. In et al., J. P., editor, *Coordination, Organizations, Institutions and Norms in Agent Systems V*, volume 6069 of *LNAI*, pages 114–129. Springer.
-  Hübner, J. F., Boissier, O., Kitio, R., and Ricci, A. (2009). Instrumenting Multi-Agent Organisations with Organisational Artifacts and Agents. *Journal of Autonomous Agents and Multi-Agent Systems*.
-  Hübner, J. F., Sichman, J. S., and Boissier, O. (2002). A model for the structural, functional, and deontic specification of organizations in multiagent systems. In Bittencourt, G. and Ramalho, G. L., editors, *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02)*, volume 2507 of *LNAI*, pages 118–128, Berlin. Springer.






## Bibliography III

-  Gâteau, B., Boissier, O., Khadraoui, D., and Dubois, E. (2005). Moiseinst: An organizational model for specifying rights and duties of autonomous agents. In *Third European Workshop on Multi-Agent Systems (EUMAS 2005)*, pages 484–485, Brussels Belgium.
-  Gutknecht, O. and Ferber, J. (2000). The MadKit agent platform architecture. In *Agents Workshop on Infrastructure for Multi-Agent Systems*, pages 48–55.
-  Hannoun, M., Boissier, O., Sichman, J. S., and Sayettat, C. (2000). MOISE: An organizational model for multi-agent systems. In Monard, M. C. and Sichman, J. S., editors, *Proceedings of the International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI (IBERAMIA/SBIA'2000)*, Atibaia, SP, Brazil, November 2000, *LNAI* 1952, pages 152–161, Berlin. Springer.






## Bibliography V

-  Hübner, J. F., Sichman, J. S., and Boissier, O. (2006). S-MOISE+: A middleware for developing organised multi-agent systems. In Boissier, O., Dignum, V., Matson, E., and Sichman, J. S., editors, *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *LNCS*, pages 64–78. Springer.
-  Hübner, J. F., Sichman, J. S., and Boissier, O. (2007). Developing Organised Multi-Agent Systems Using the MOISE+ Model: Programming Issues at the System and Agent Levels. *Agent-Oriented Software Engineering*, 1(3/4):370–395.
-  Malone, T. W. (1999). Tools for inventing organizations: Toward a handbook of organizational process. *Management Science*, 45(3):425–443.



## Bibliography VI

-  Morin, E. (1977).  
*La méthode (1) : la nature de la nature.*  
Points Seuil.
-  Okuyama, F. Y., Bordini, R. H., and da Rocha Costa, A. C. (2008).  
A distributed normative infrastructure for situated multi-agent organisations.  
In Baldoni, M., Son, T. C., van Riemsdijk, M. B., and Winikoff, M., editors, *DALT*, volume 5397 of *Lecture Notes in Computer Science*, pages 29–46. Springer.
-  Ossowski, S. (1999).  
*Co-ordination in Artificial Agent Societies: Social Structures and Its Implications for Autonomous Problem-Solving Agents*, volume 1535 of *LNAI*.  
Springer.





## Bibliography VII

-  Pianti, M., Ricci, A., Boissier, O., and Hubner, J. (2009a).  
Embodying organisations in multi-agent work environments.  
*In IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2009)*, Milan, Italy.
-  Pianti, M., Ricci, A., Boissier, O., and Hübner, J. F. (2009b).  
Embodied organisations in mas environments.  
In Braubach, L., van der Hoek, W., Petta, P., and Pokahr, A., editors, *Proceedings of 7th German conference on Multi-Agent System Technologies (MATES 09)*, Hamburg, Germany, September 9-11, volume 5774 of *LNCS*, pages 115–127. Springer.
-  Pynadath, D. V. and Tambe, M. (2003).  
An automated teamwork infrastructure for heterogeneous software agents and humans.  
*Autonomous Agents and Multi-Agent Systems*, 7(1-2):71–100.



## Bibliography VIII

-  Ricci, A., Pianti, M., Viroli, M., and Omicini, A. (2009).  
Environment programming in CArtAgO.  
*In Multi-Agent Programming: Languages, Platforms and Applications, Vol.2*. Springer.
-  Tambe, M. (1997).  
Towards flexible teamwork.  
*Journal of Artificial Intelligence Research*, 7:83–124.

