Organisation Oriented Programming
with $\text{MOISE}^+$
at the system and agent levels

Jomi F. Hübner
(collaboration with Olivier Boissier and Jaime S. Sichman)

ENS Mines Saint Etienne, France
hubner@emse.fr

LIP6 Seminars
Outline

1. Context
2. MOISE$^+$
3. System level
4. Agent level
5. Summary
Reasons for organisation in MAS
‘normative view’

- Multi-agent systems have two properties which seem contradictory:
  - a **global** purpose × **autonomous** agents

  While the autonomy of the agents is essential for MASs, it may cause loss in the global coherence of the system

- The **organisation** of a MAS is used to solve this problem
  **constraining** the agents’ behaviour towards global purposes

- For example, when an agent adopts a role, it adopts a set of behavioural constraints that support a global purpose
Constraining the agents’ autonomy by **Norms** mechanisms

- **Regimented** norms: the organisation prevents their violation by the agents
  - e.g. messages that do not follow the protocol are discarded

- **Enforced** norms: agents decide to obey or not to them, the organisation lets the agents the possibility to violate them
  - e.g. a master thesis should be written in two years
  - Detection of violations, decision about sanctions
Reasons for organisation in MAS
‘constitutive view’

The organisation helps the agents to cooperate by defining common
- global tasks
- protocols

For example, ‘to bid’ for a product on eBay is an institutional action only possible because the eBay defines the rules for that very action
- the bid protocol is a constraint but it also creates the action
Programming organised MAS

- **System approach:**
  - Develop an organisational infrastructure that helps the agents to participate in the organisation.
  - Develop an organisational infrastructure that ensures or enforce that the organisational norms will be followed.
    - The agents have to respect the organisation despite their architecture.

- **Agent-centred approach:**
  - Develop agent reasoning mechanisms that are aware of the organisation.
  - Not suitable for all kinds of open systems (unknown agents may not behave well!)
1. Context

2. Moise$^+$
   - General view
   - Example
   - Software

3. System level

4. Agent level

5. Summary
MOISE$^+$ – general view

- **Organisation Modelling Language (OML)**
  - $\Rightarrow$ allows the designer to specify the organisation of a MAS along three dimensions (structural, functional, deontic)

- **Organisational Infrastructure**
  - $\Rightarrow$ interprets the OML and then constraints/supports the agents in the specified organisation
    - by means of regimentation, enforcement, tools for cooperative tasks, ...
    - allows agents to interact with the organisation (agent programming issues)

- **Support for agent programming**
MOISE$^+$ – general view

- Organisation Modelling Language (OML)
  → allows the designer to specify the organisation of a MAS along three dimensions (structural, functional, deontic)

- Organisational Infrastructure
  → interprets the OML and then constraints/supports the agents in the specified organisation
    - by means of regimentation, enforcement, tools for cooperative tasks, ...
    - allows agents to interact with the organisation (agent programming issues)

- Support for agent programming
Moise$^+$ – general view

- Organisation Modelling Language (OML)
  - allows the designer to specify the organisation of a MAS along three dimensions (structural, functional, deontic)

- Organisational Infrastructure
  - interprets the OML and then constraints/supports the agents in the specified organisation
    - by means of regimentation, enforcement, tools for cooperative tasks, ...
    - allows agents to interact with the organisation (agent programming issues)

- Support for agent programming
Moise+ by example: ‘writing a paper’

Structural Specification

author

writer

editor

1..5

1..1

wpgroup
**MOISE⁺ by example: ‘writing a paper’**

**Functional Specification**
**MOISE** by example: ‘writing a paper’

**Missions**
Moise⁺ by example: ‘writing a paper’

Permissions
MOISE by example: ‘writing a paper’

Obligations

Organisational Programming – J. Hübner
LIP6, Paris, Jan 2009
Organisational infrastructures
- $S$-$\text{MOISE}^+$ (‘traditional’ approach)
- ORA4MAS (approach based on artifacts)

Agent programming
- $J$-$\text{MOISE}^+$ (BDI agent with \textit{Jason} language)
1. Context

2. MOISE+

3. System level
   - S-MOISE+
   - A&A
   - ORA4MAS

4. Agent level

5. Summary
Organisational Infrastructure (for open systems)

$S$-$\text{MOISE}^+$, ISLANDER, STEAM, ....
**Two main components:** OrgManager and OrgBox
The OrgBox is the interface that the agents use to access the organisational layer and thus the communication layer too.

OrgBox must be used to:
- Change the organisational entity (adopt a role, for instance)
- Send a message to another agent
- Get the organisational entity state
  - However, only a personalised version of the entity is given from OrgManager to OrgBox, respecting the acquaintance relation

OrgManager notifies an agent’s OrgBox about every change in the state of a scheme to which the agent has committed to.

No particular agent architecture is required.
OrgManager

- Maintains the current state of the organisational entity
  - Created groups and schemes
  - Role assignments (Agents to Roles)
  - Mission assignments (Agents to Missions)
  - Change goal states (satisfied or not)
  - ...

- Maintains the current state of the organisational specification

- Receives messages from the other agents’ OrgBoxes asking for changes in the organisational entity/specification

- Regiments some norms
Organisational entity dynamics

The organisational entity is changed by requests coming from agents’ OrgBoxes.

Examples of messages:

- `create_group("g1", "wpgroup")`: a group called `g1` is created using the ‘wpgroup’ group specification.
- `create_scheme("wp", "g1")`: an instance of the ‘wp’ scheme specification is created; the agents in group `g1` are responsible for this scheme’s missions.
- `adopt_role("bob", "editor", "g1")`: the agent ‘bob’ wants to adopt the role ‘editor’ in group ‘g1’.
- …
Regimentation of an organisational action
e.g. role adoption

The adoption of a role $\rho$ by an agent $\alpha$ in group $g$ has the following constraints:

- The role $\rho$ must belong to the specification of group $g$.
- The number of $\rho$ players in $g$ must be less than or equal to the maximum number of $\rho$ players defined in the specification of group $g$.
- For all roles $\rho_i$ that agent $\alpha$ already plays in $g$, the roles $\rho$ and $\rho_i$ must be compatible in the specification of group $g$. 
Goal’s state and coordination

When an agent is committed to a mission, it is responsible for a number of goals. Only some of them might be possible at a given moment (those whose pre-condition goals are already satisfied).
Goal’s state and coordination

When an agent is committed to a mission, it is responsible for a number of goals. Only some of them might be possible at a given moment (those whose pre-condition goals are already satisfied).
Goal’s state and coordination

When an agent is committed to a mission, it is responsible for a number of goals. Only some of them might be possible at a given moment (those whose pre-condition goals are already satisfied).
Goal’s state and coordination

When an agent is committed to a mission, it is responsible for a number of goals. Only some of them might be possible at a given moment (those whose pre-condition goals are already satisfied).
Goal’s state and coordination

When an agent is committed to a mission, it is responsible for a number of goals. Only some of them might be possible at a given moment (those whose pre-condition goals are already satisfied).
## Useful tools — **MOISE^+ GUI**

### Players

- **jaime** committed to mManager
- **jomi** committed to mColaborator
- **olivier** committed to mColaborator
- **olivier** committed to mBib

### Table

<table>
<thead>
<tr>
<th>goal</th>
<th>state</th>
<th>committed</th>
<th>arguments</th>
<th>plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>wplib</td>
<td>waiting</td>
<td>[jaime]</td>
<td></td>
<td>= fdv, sv</td>
</tr>
<tr>
<td>fdv</td>
<td>possible</td>
<td>[jaime]</td>
<td></td>
<td>= wtitle, wabs, wsectitles</td>
</tr>
<tr>
<td>wtitle</td>
<td>achieved : [jaime]</td>
<td>[jaime]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wabs</td>
<td>achieved : [jaime]</td>
<td>[jaime]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wsectitles</td>
<td>achieved : [jaime]</td>
<td>[jaime]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sv</td>
<td>achieved</td>
<td>[]</td>
<td></td>
<td>= wsecs, finish</td>
</tr>
<tr>
<td>wsecs</td>
<td>achieved : [jomi, olivier]</td>
<td>[olivier, jomi]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>finish</td>
<td>achieved</td>
<td>[]</td>
<td></td>
<td>= wconc</td>
</tr>
<tr>
<td>wconc</td>
<td>achieved : [jaime]</td>
<td>[jaime]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Motivations for another approach

- Organisational services are implemented as ‘special’ agents — which are conceptually different — agents doing services

- Organisational decisions are taken by the organisational infrastructure — the organisational infrastructure has too much power
  
  - For example, if some agent performs a forbidden action, the middleware detects it as a violation and decides to apply a sanction (or even disable the execution of the action)

services taken decisions which are ‘closed’ for the agents
**A&A model**

Artifacts, Agents, Workspaces [Ricci *et al.* 07]
Artifact model
Organisational artifacts in ORA4MAS

- based on A&A and $\text{MOISE}^+$
- agents create and handle organisational artifacts
- artifacts in charge of regimentations, detection and evaluation of norms compliance
- agents are in charge of decisions about sanctions
Organisational artifacts in **ORA4MAS**

- based on A&A and **MOISE**
- agents create and handle organisational artifacts
- artifacts in charge of regimentations, detection and evaluation of norms compliance
- agents are in charge of decisions about sanctions
Reputation Artifact in ORA4MAS

- Instrument to help in the enforcement of norms
- Indirect sanction system
- Considers the public character of the reputation process
- Publish an evaluation of the agents from the organisation point of view
1. Context

2. MOISE$^+$

3. System level

4. Agent level
   - $J$-MOISE$^+$
   - actions
   - events
   - goals
   - example

5. Summary
$\mathcal{J}$-MOISE$^+$: *Jason* + *MOISE*$^+$

- **S-MOISE$^+$** and ORA4MAS provides general services for the agents to be organised, but does not help us to program the agents or the agents’ reasoning about its organisation.

- **$\mathcal{J}$-MOISE$^+$**
  - Programming agents with AgentSpeak
  - BDI agents (reactive planning) – higher abstraction level
  - Help the programmer to determine when the agent should adopt a role, a mission, ...
  - Enable the agents to access organisational information
  - Independence from the distribution/communication layer
  - Using *Jason*, an open-source interpreter for a variant of AgentSpeak, developed by Rafael Bordini and Jomi Hübner
General view

Jason Agent

Belief Base
Plan Library
Intentions

J-Moise+
Organisation Awareness mechanism

Organisational Infrastructure

Context MOISE+ System level Agent level Summary MOISE+ J-Moise+ actions events goals example

Organisational Programming – J. Hübner LIP6, Paris, Jan 2009
Organisational actions in AgentSpeak I

Example (AgentSpeak plan)

```
+some_event : some_context
<- jmoise.create_group(wpgroup).
```

Some available Organisational Actions:

- For groups:
  - `create_group(<GrSpecId>[,<SuperGrId>])`
  - `remove_group(<GrId>)`

- For schemes:
  - `create_scheme(<SchSpecId> [,<groups>])`
  - `add_responsible_group(<SchId>,<GrId>)`
  - `remove_scheme(<SchId>)`
  - `set_goal_state(<SchId>,<Goal>,<State>)`
Organisational actions in AgentSpeak II

- **For Agents:**
  - `adopt_role(<RoleId>,<GrId>)`
  - `remove_role(<RoleId>,<GrId>)`
  - `commit_mission(<MisId>,<SchId>)`
  - `remove_mission([<MisId>,] <SchId>)`

- Those actions 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
Handling organisational **events** in AgentSpeak

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

**Example (A new group is created)**

```prolog
+group(wpgroup,GId) : true
    <- jmoise.adopt_role(editor,GId).
```

or

```prolog
+group(wpgroup,GId)[owner(O)] : my_friend(O)
    <- jmoise.adopt_role(editor,GId).
```

**Example (Some group is destroyed)**

```prolog
-group(wpgroup,GId) <- .print("Group removed!").
```
Available organisational events I

- +/- group(\(<\text{GrSpecId}>,<\text{GrId}>\))
  
  [owner(\(<\text{AgName}>)\), super\_gr(G)]:
  perceived by all agents when a group is created (event +) or removed (event −) by AgName

- +/- play(\(<\text{AgName}>, <\text{RoleId}>, <\text{GrId}>\)):
  perceived by the agents of GrId when an agent adopts (event +) or remove (event −) a role in group GrId

- +/- commitment(\(<\text{AgName}>, <\text{MisId}>, <\text{SchId}>\)):
  perceived by the SchId players when an agent commits or removes a commitment to a mission MisId in scheme SchId
Available organisational events II

- \( \pm \text{scheme}(\langle \text{SchSpecId} \rangle, \langle \text{SchId} \rangle) \)
  
  \[ \text{[owner}(\langle \text{AgName} \rangle)]: \]
  
  perceived by all agents when a scheme is created (+) or finished (-) by \text{AgName}

- \( \text{+ scheme\_group}(\langle \text{SchId} \rangle, \langle \text{GrId} \rangle) \):
  
  perceived by \text{GrId} players when this group becomes responsible for the scheme \text{SchId}

- \( \text{+ goal\_state}(\langle \text{SchId} \rangle, \langle \text{GoalId} \rangle, \langle \text{State} \rangle) \):
  
  perceived by \text{SchId} players when the state of some goal changes
Available organisational events III

- + obligation(<SchId>, <MisId>)
  [role(<RoleId>), group(<GrId>)]
  perceived by an agent when it has an organisational obligation for a mission. It has a role (RoleId) in a group (GrId) responsible for a scheme (SchId) and this role is obligated to a mission in this scheme

- + permission(<SchId>, <MisId>)
  [role(<RoleId>), group(<GrId>)]
Achieving organisational goals

An achievement goal event (+!g) is create when an organisational goal g is possible

Example (Organisational goal)

If an agent is committed to a mission with goal wsec, whenever this goal is possible (all its pre-condition goals are satisfied), the following plan may be selected:

+!wsec[scheme(Sch)]
  : commitment(A, mBib, Sch)
  <- ..... actions to write the section .....;
    .send(A,tell,[references]);
    jmoise.set_goal_state(Sch, wsec, satisfied).

The context of this plan uses organisational information to constrain its execution.
Example: Writing paper
Organisation Specification

```xml
<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>
   </structural-specification>
</organisational-specification>
```
Execution sample I

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)
Execution sample II

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)
Execution sample III

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)
Execution sample IV

all  perception:  goal_state(s1,*,unsatisfied)

jaime  (only wtitle is possible, Jaime should work)
  event:  +!wtitle
  action:  jmoise.set_goal_state(s1,wtitle,satisfied)
Execution sample V

jaime event: +!wabs

action: jmoise.set_goal_state(s1,wabs,satisfied)
Execution sample VI

jaime event: +!wsectitles
action: jmoise.set_goal_state(s1,wsectitles,satisfied)
Execution sample VII

olivier, jomi  
**event:**  $+$!wsecs

**action:**  `jmoise.set_goal_state(s1,wsecs,satisfied)`
Execution sample VIII

```
jaime  event:  +!wcon; ...

olivier  event:  +!wref; ...
```
Execution sample IX

all  action: jmoise.remove_mission(s1)

jaime  action: jmoise.jmoise.remove_scheme(s1)
Useful tools — Mind inspector

```
play(gaucho1,herder,gr_herding_grp_13)[source(orgManager)];
play(gaucho4,herdboy,gr_herding_grp_13)[source(orgManager)];
play(gaucho5,herdboy,gr_herding_grp_13)[source(orgManager)];
pos(45,44,128)[source(percept)].
scheme(herd_sch,sch_herd_sch_18)[owner(gaucho3),source(orgManager)].
scheme(herd_sch,sch_herd_sch_12)[owner(gaucho1),source(orgManager)].
scheme_group(sch_herd_sch_12,gr_herding_grp_13)[source(orgManager)].
steps(700)[source(self)].
target(6,44)[source(gaucho1)].
```

**Rules**

```
random_pos(X,Y) :-
  (pos(AgX,AgY,-418) & (jia.random(RX,40) & ((RX > 5) & ((X = ((RX-20)+AgX)) & ((X >
```

**Intentions**

<table>
<thead>
<tr>
<th>Sel</th>
<th>Id</th>
<th>Pen</th>
<th>Intended Means Stack (hide details)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16927</td>
<td></td>
<td>suspended-self</td>
<td>+!be_in_formation[scheme(sch_herd_sch_12),mission(help)]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+!be_in_formation[scheme(Sch),mission(Mission)]</td>
</tr>
</tbody>
</table>

Organisational Programming — J. Hübner

LIP6, Paris, Jan 2009
Summary — **S-MOISE⁺**

- Ensures that the agents follow some of the norms specified for the organisation (cardinality of groups, communication and acquaintance links, role and mission adoption, goal satisfaction)
- The organisation is *interpreted at runtime*, it is not hardwired in the agents code
- It has a synchronisation mechanism for scheme execution
- It is suitable for open systems as no specific agent architecture is required
- An implementation is available at [http://moise.sourceforge.net](http://moise.sourceforge.net)
Summary — ORA4MAS

- Same services of $S\text{-MOISE}^+$
- Based on artifacts that agents can handle
  (non-autonomous part of the system)
- on going work — Rosine Kitio
Summary — \textit{J-MOISE}⁺

- Supports the development of organised agents using
  - Logic-based language
  - BDI architecture
  - AgentSpeak agent-oriented programming language
  - Declarative and goal oriented programming
  - Meta-programming
    \texttt{.drop\_intention\(_\{\quad [\text{role(writer)}]\}\)}

- Approach based on
  - Organisational actions, events, and goals

- But, it is ‘just’ an integration,
  it still lacks organisational reasoning
  (ongoing work [Cosmin])
Summary — J-MOISE$^+$

- Supports the development of organised agents using
  - Logic-based language
  - BDI architecture
  - AgentSpeak agent-oriented programming language
  - Declarative and goal oriented programming
  - Meta-programming
  \[
  \text{.drop_intention(\_\[role(writer)\])}
  \]

- Approach based on
  - Organisational actions, events, and goals

- But, it is ‘just’ an integration,
  it still lacks organisational reasoning
  (ongoing work [Cosmin])
More information

- http://moise.sf.net
- http://jason.sf.net

Points of view on organisation

emergent organisation

(a) type AR

observer

representation of observed organisation
Points of view on organisation

<table>
<thead>
<tr>
<th>emergent organisation</th>
<th>(a) type AR</th>
</tr>
</thead>
<tbody>
<tr>
<td>observer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>pre-defined organisation</td>
<td>(c) type OR</td>
</tr>
<tr>
<td>observer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>organisation</td>
<td></td>
</tr>
</tbody>
</table>

- □ representation of observed organisation
- △ representation of institutional organisation
## Points of view on organisation

<table>
<thead>
<tr>
<th>Emergent organisation</th>
<th>Agents unable to represent their organisation</th>
<th>Agents able to represent their organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Type AR</td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Pre-defined organisation</td>
<td>(c) Type OR</td>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **Observer**
- **Representation of observed organisation**
- **Δ** Representation of institutional organisation
# Points of view on organisation

<table>
<thead>
<tr>
<th>Emergent Organisation</th>
<th>Agents unable to represent their organisation</th>
<th>Agents able to represent their organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Type AR</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>Pre-defined Organisation</td>
<td>(c) Type OR</td>
<td>(d) Type OC</td>
</tr>
<tr>
<td><img src="image3" alt="Diagram" /></td>
<td><img src="image4" alt="Diagram" /></td>
<td><img src="image5" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **[Representation of observed organisation](#)**
- **[Representation of institutional organisation](#)**

---

Organisational Programming – J. Hübner  
LIP6, Paris, Jan 2009