Programming MAS reorganisation with $\text{MOISE}^+$

Jomi F. Hübner$^1$  Olivier Boissier$^2$  Jaime S. Sichman$^3$

$^1$Department of Computer Science
University of Blumenau

$^2$Multi-Agent Systems
G2I ENS Mines Saint-Etienne

$^3$Intelligent Techniques Laboratory
University of São Paulo

Dagstuhl Seminar – Foundations and Practice of Programming
Multi-Agent Systems, 2006
Outline

1. Organisation
   - Context
   - MOISE+

2. Reorganisation
   - Group
   - Phases

3. Programming with (re)organisation
   - Requirements
   - S-MOISE+
   - J-MOISE+

4. Summary and Future Work
A multiagent system has two properties which seems controversial:

- a **global** purpose × **autonomous** agents

While the autonomy of the agents is essential for the MAS, it may cause the looseness of the global congruence.

- The **organisation** of an MAS is used to solve this conflict constraining the agents’ behaviour towards global purposes.

- Example: when an agent adopts a role, it indeed adopts a set of behavioural constraints that collaborates for a global purpose.
Our point of view on organisation

agents' behavior space
Our point of view on organisation

- Environment
- Agents' behavior space

Hübner, Boissier, Sichman: Moise⁺, S-Moise⁺, J-Moise⁺
Our **point of view** on organisation

Roles, groups, communication links, authority links, ...

E.g.: AGR [Ferber and Gutknecht, 1998],

Hübner, Boissier, Sichman  |  Moise⁺, S-Moise⁺, J-Moise⁺
Our point of view on organisation

Goals, plans, missions, norms, ...

e.g.: TÆMS [Decker, 1998]

Hübner, Boissier, Sichman
Our point of view on organisation

e.g.: tove, Opera, Steam
The problem of finding a good organisation 1

- The organisation does not lead to global purpose.
The problem of finding a good organisation II

- The organisation extinguish the agents’ autonomy.
A good organisation

- Not so narrow neither so tolerant.
- Initially, the problem of finding a good organisation can be solved by the MAS designer.
- In **dynamic** and **open** environments, the agents themselves must change its organisation.
  - **reorganisation**
- Thus we need an organisational model suitable for reorganisation: $\textit{MOISE}^+$. 
A proposal to join roles (structure) and plans (functioning) with some **independence** between them to simplify reorganisation. The $\text{MOISE}^+$ is structured along three levels:

1. **Individual** level: definition of the organisation’s roles.
2. **Social** level: definition of interconnections between roles that constraint the agent behaviour
   - related to other agents (e.g. authority, communication channels),
   - related to common task (e.g. commitments).
3. **Collective** level: the aggregation of roles in large structures.
Study Case: Robocup small size league 1
Study Case: Robocup small size league II

organisation
to play soccer

soccer agents

Hübner, Boissier, Sichman

Moise\textsuperscript{+}, S-Moise\textsuperscript{+}, J-Moise\textsuperscript{+}
Specifying the JOJTEAM organisation: structure I

- soc
- player
- back
- left
- right
- attacker
- goalkeeper
- team
- ReorgGr

Structure
- inheritance: min..max
- composition: role
- sub-groups scope: group

- links
- intra-group
- inter-group
- acq
- com
- aut
- compat

Hübner, Boissier, Sichman
Moise⁺, S-Moise⁺, J-Moise⁺
Specifying the JOJTEAM organisation: structure II

Organizational Entity

- Dida: goalkeeper
- Lucio: back
- Juan: leader
- Cafu: leader
- Kaka: middle
- Emerson: middle
- Ze Roberto: attacker
- Ronaldinho: attacker
- Roberto Carlos: attacker
- Ronaldo: attacker
- Adriano: attacker
Specifying the JOJTEAM organisation: functioning

Scheme
- Missions
  - Goal

Roles and Missions

<table>
<thead>
<tr>
<th>Role</th>
<th>Deontic</th>
<th>Mission</th>
</tr>
</thead>
<tbody>
<tr>
<td>mCGback</td>
<td>obligation</td>
<td>mKG</td>
</tr>
<tr>
<td>left</td>
<td>obligation</td>
<td>mCG</td>
</tr>
<tr>
<td>right</td>
<td>obligation</td>
<td>mCG</td>
</tr>
<tr>
<td>attacker</td>
<td>obligation</td>
<td>mCG</td>
</tr>
<tr>
<td>goalkeeper</td>
<td>obligation</td>
<td>mBG</td>
</tr>
</tbody>
</table>
Specifying the JOJTEAM organisation: functioning II

m1, m2, m3

score a goal

m1, m2, m3

got the ball

m1

get the ball

m1

go towards the opponent field

m1

be placed in the middle field

m2

be placed in the opponent goal area

m3

be placed in the middle field

m2

kick the ball to the goal area

m2

kick the ball to the goal area

m2

go to the opponent back line

m1

shot at the opponent’s goal

Key

Scheme missions

goal success rate

sequence choice parallelism

Organizational Entity

Lucio ---- m1

Cafu ------ m2

Rivaldo ----- m3

Hübner, Boissier, Sichman

Moise+, S-Moise+, J-Moise+
Approach to reorganise the team

i) Create a special group of agents specialised in reorganisation.

ii) This new group is also organised.

iii) Since the soccer agents follow the organisation, the new organisation is easily implemented.
Structural dimension of the reorganisation

- OrgManager
- Reorg
- Monitored
- Designer
- Selector
- Monitor
- ReorgExpert
- Historian
- OrgParticipant
- soc

Hübner, Boissier, Sichman
Moise⁺, S-Moise⁺, J-Moise⁺
Functional dimension of the reorganisation

Deontic relations:

- OrgManager $\rightarrow$ obl($m_1$)
- Monitor $\rightarrow$ obl($m_2$)
- ReorgExpert $\rightarrow$ obl($m_4$)
- OrgParticipant $\rightarrow$ per($m_5$)
- Selector $\rightarrow$ obl($m_6$)
Example of **Monitoring** goal 1

- **JOJTEAM**: the Monitor agent starts a reorganisation with some frequency (5 reorganisation each game)
**Design goal 1**

- **JOJTEAM**: 9 designers that always propose the same kind of reorganisation (1×1×3, 4×1, increase the players area, change the team goals, ...)

![Diagram with nodes and edges representing the design goal and its associated agents and roles.]

**Hübner, Boissier, Sichman**

*Moise⁺, S-Moise⁺, J-Moise⁺*
The reorganisation change must be proposed as a reorganisation plan.

Example:

1. remove all roles from group team;
2. create role back extending player;
3. set back property area as "-137x40 10x-40";
4. add role back into group team;
5. define mission mKG as \{kickToGoal\};
6. add mission mKG as obligation for back;

... 

A plan may change either the structure or the functioning (e.g. add a new mission for the Goalkeeper).
Selection goal

- **JOJTeam**: an agent that uses Q-Learning to learn when to choose each designer proposal
- **State**: match time (5 moments) and game score (-2,-1,0,1,2)
- **Actions**: choose designer 1, choose designer 2, ..., choose designer 9
- **Reward**: goals
The OrgManager agent executes the reorganisation plan selected.
Results

![Graph showing final score vs. game number]

-5 -4 -3 -2 -1 0 1 2 3

0 500 1000 1500 2000

final score
game number

Hübner, Boissier, Sichman, Moise\(^+\), S-Moise\(^+\), J-Moise\(^+\)
Programming organised agents

- How to implement MAS that follow an organisation?
- 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!).
- Organisational approach (our focus):
  - Develop a multi-agent infrastructure that ensures that the organisational constraints will be followed?
  - The agents have to respect the organization despite their architecture.
- Available tools:
  - Ameli [Esteva et al., 2004] (based on Islander)
  - MadKit [Gutknecht and Ferber, 2000] (based on Agr)
  - Karma [Pynadath and Tambe, 2003] (based on Steam)
- These tools are not conceived for reorganisation.
Two main components: OrgManager and OrgBox.
The OrgBox is the interface that the agents use to access the organizational layer and thus the communication layer.

**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 to respect 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 Agent

- Maintains the current state of the organisational entity
  - Created groups and schemes
  - Role assignments (Agents to Roles)
  - Mission assignments (Agents to Missions)
  - Change goals state (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.
- Ensures that an agent request is allowed by the organisation.
The entity is changed by requests coming from agents’ OrgBoxes. Examples of messages:

- `createGroup("g1","team")`: a group called `g1` is created from the “team” group specification.

- `createSubGroup("d1", "defense", "g1")`: a group called `d1` is created from the “defense” specification as a `g1` sub-group.

- `createScheme("side_attack", "g1")`: an instance of the “side_attack” scheme specification is created, the agents of the group `g1` are responsible for these scheme’s missions.

- `adoptRole("Cafu", "leader", "d1")`: the agent “Cafu” wants to adopt the role “leader” in group “d1”.

- ....
Role adoption

The adoption of a role $\rho$ by an agent $\alpha$ in the 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 lesser or equals than 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 $g$. 
Permitted goals and agent coordination for scheme execution

When an agent is committed to a mission, it is responsible for some goals. Only some of them may be permitted (those whose pre-goals are already satisfied).

```
    score a goal
     /         \
    /           \
   /             \n/m1, m2, m3
     |             |
    /   \         /
/m1   / \       / m3
get the ball

   /  \  \
/m1  / \   \ m3
/go towards the opponent field

   /  \  \
/m2  / \   \ m2
be placed in the middle field

   /  \  \
/m3  / \   \ m2
be placed in the opponent goal area

   /  \  \
/m2  / \   \ m2
be placed in the opponent back line

   /  \  \
/m3  / \   \ m1
kick the ball to (agent committed to m2)
```
Permitted goals and agent coordination for scheme execution

When an agent is committed to a mission, it is responsible for some goals. Only some of them may be permitted (those whose pre-goals are already satisfied).
When an agent is committed to a mission, it is responsible for some goals. Only some of them may be permitted (those whose pre-goals are already satisfied).
J- Моись$: Jason + Moise$+

- $S-Moise^+$ provides that organisational constraints are followed, but does not help us to program the agents or the agent reasoning about its organisation.

- $J-Moise^+$
  - Programming agents with AgentSpeak.
  - BDI agents (reactive planning) – higher abstraction level.
  - Enable the user to state when the agent should adopt a role, a mission, ...
  - Enable the agents to deal with multiple goals.
  - Enable the agents to access organisational information.
  - Independence from the distribution/communication layer.
  - Use *Jason*, an open-source interpreter of AgentSpeak, developed by Rafael Bordini and Jomi Hübner.
General view

Jason Agent

Agent Code (AgentSpeak)

Organisational Architecture

Internal Actions

Jason Communication Infrastructure

Jason Agent

Organisational Specification

OrgManager Architecture

Organisational Entity
Organisational Actions in AgentSpeak

- Example:

  ```
  +someEvent : true
  <- jmoise.createGroup(wpgroup).
  ```

- Some available Organisational Actions:
  - `createGroup(<GrSpecId>[,<GrId>])`
  - `removeGroup(<GrId>)`
  - `startScheme(<SchSpecId>)`
  - `finishScheme(<SchId>)`
  - `adoptRole(<RoleId>,<GrId>)`
  - `removeRole(<RoleId>,<GrId>)`
  - `commitToMission(<MisId>,<SchId>)`
  - `removeMission([<MisId>,] <SchId>)`
Handling Organisational Events in AgentSpeak

Whenever something changes in the organisation, the organisation architecture updates the agent belief base accordingly.

- A new group is created
  +group(defense,Id) : true
  <- jmoise.adoptRole(back,Id).

  or
  +group(defense,Id)[owner(0)] : myFriend(0)
  <- jmoise.adoptRole(back,Id).

- Some group is destroyed
  -group(defense,Id) : true
  <- .print("The group ",Id," was removed!").
Available Organisational Events I

- +/- group(<GrSpecId>,<GrId>)[owner(<AgName>)]: perceived by all agents when a group is created (event +) or removed (event -) by AgName.

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

- +/- commitment(<AgName>, <MisId>, <SchId>): perceived by the SchId players when an agent commits or removes a commitment to a mission MisId in scheme SchId.

- +/- scheme(<SchSpecId>,<SchId>)[owner(<AgName>)]: perceived by all agents when a scheme is created (+), finished (-), or aborted (-) by AgName.
Available Organisational Events II

- schemeGroup(<SchId>,<GrId>): perceived by GrId players when this group becomes responsible for the scheme SchId.

- obligation(<SchId>, <MisId>)[role(<RoleId>), group(<GrId>)]: perceived by an agent when is 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.
Achieving Organisational Goals

An achievement goal event is create when an organisational goal is permitted.

- Example: if an agent is committed to a mission with goal “kickToGoal”, when this goal is permitted (all its pre-goals are satisfied), the following plan is selected:

  ```
  +!kickToGoal[scheme(Sch)]: true
  <- ?goodLocationToKick(X,Y);
  !carryBallTo(X,Y);
  kick;
  jmoise.setGoalState(Sch, kickToGoal, satisfied).
  ```

- Using organisational information:

  ```
  +!kickToGoal[scheme(S)]: commitment(lucio, m2, Sch)
  <- ....
  ```
The $\text{MOISE}^+$ organisational model supports the specification of an MAS’s organisation which intends to reorganise itself

- Since the reorganisation is a process like any other, an agent that understand $\text{MOISE}^+$ specification can participate in the reorganisation — thus it simplifies openness, “team programming”.
- The reorganisation can have many monitoring and designing strategies.
- The reorganisation plans simplifies the design of new organisation and deal with some implementation problems.
- The $\text{MOISE}^+$ independence between struncture and functioning simplifies the construction of reorganisation plans.
Summary II

- **S-Moise⁺**: Ensures that the agents follow some of the constraints 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 since no specific agent architecture is required.

- An implementation is available at http://moise.sourceforge.net
Summary III

- \textit{J-Moise}^+
  - Program agents ("ordinary" or re-organisational) with
    - Logic
    - BDI
    - AgentSpeak
  - Proposal based on
    - OrgManager
    - Organisational actions
    - Organisational events

- An implementation is available at http://jason.sourceforge.net

Hübner, Boissier, Sichman \textit{Moise}^+, S-Moise^+, J-Moise^+
Further work

- Although implemented for $\text{MOISE}^+$ organisational model, some ideas could be adapted for other models:
  - Common organisational ontology

- Implementation of a sanction system to deal with agents that do not achieve their organisational goals (Moise-inst [Gateau 04])

- Development of an agent internal mechanism to deal with organisational aspects

- Organisational reasoning.

- Development of tools to edit organisation, generate code, ...
References


References II

An organizational ontology for enterprise modeling.

The MadKit agent platform architecture.

A model for the structural, functional, and deontic specification of organizations in multiagent systems.

Using the \textsc{Moise}$^+$ for a cooperative framework of MAS reorganisation.

\textsc{S-Moise}$^+$: A middleware for developing organised multi-agent systems.
An automated teamwork infrastructure for heterogeneous software agents and humans.

Building dynamic agent organizations in cyberspace.
*IEEE Internet Computing, 4*(2).