Programming MAS reorganisation with $\mathcal{M}$OISE$^+$

Jomi F. Hübner[1]   Olivier Boissier[2]   Jaime S. Sichman[3]

[1]Department of Computer Science
University of Blumenau
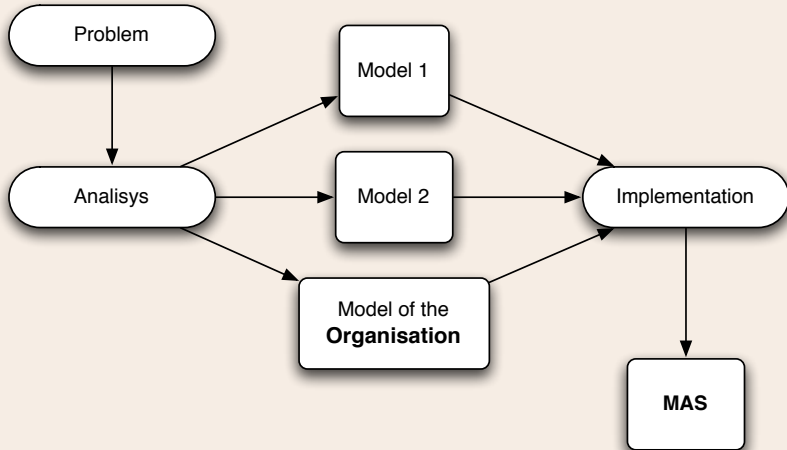
[2]Multi-Agent Systems
G2I ENS Mines Saint-Etienne

[3]Intelligent Techniques Laboratory
University of São Paulo

SBES Workshop on
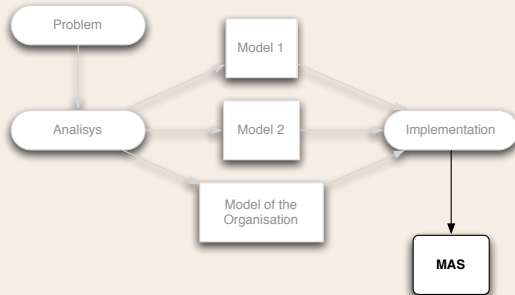Software Engineering for Agent-oriented Systems, 2006

Outline

Organisation in MAS: **Modelling** I
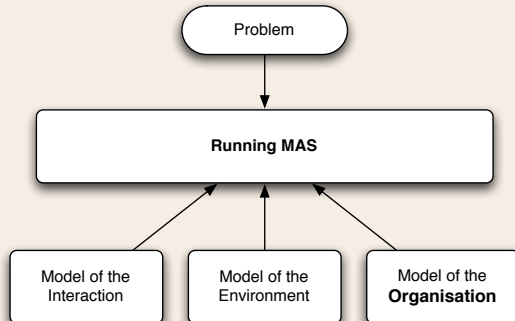
Organisation in MAS: **Modelling** II

- The organisation model is not available to the agents.

- Not suitable for open systems.

- No organisational autonomy.

- Runtime reorganisation is not conceived.

Organisation in MAS: Modelling & **Service** I

From closed to open
systems:

- Agents arrive and leave
  dynamically.

- Agents behaviour may
  change dynamically due
  to environment evolving
  and learning.

- The organisation model
  is available to these
  agents.

- Runtime reorganisation
  is possible (modelling
  by the agents!).

```
        ┌─────────────┐
        │   Problem   │
        └─────────────┘
               │
               ▼
┌───────────────────────────────┐
│        Running MAS            │
└───────────────────────────────┘
     ▲         ▲         ▲
┌──────────┐ ┌──────────┐ ┌──────────┐
│Model of  │ │Model of  │ │Model of  │
│the       │ │the       │ │the       │
│Interaction│ │Environment│ │Organisation│
└──────────┘ └──────────┘ └──────────┘
```

Organisation in MAS: Modelling & **Service** II

- A multi-agent system has two properties which seems controversial:
  - a global purpose $\times$ 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
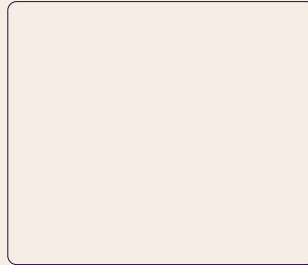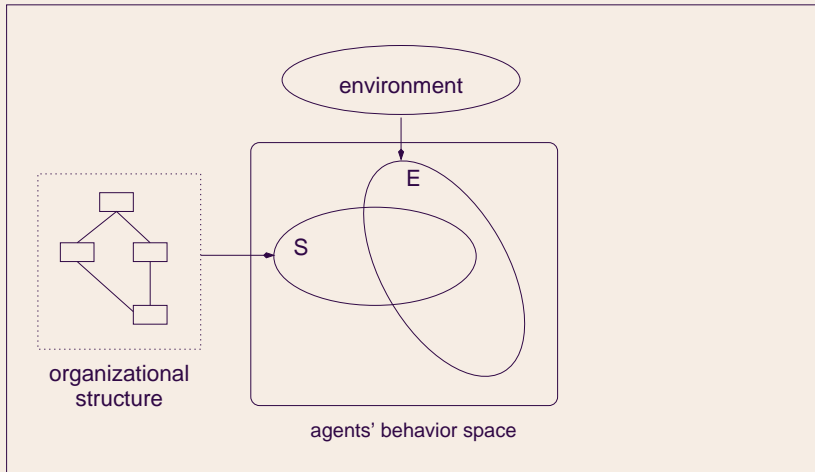

agents' behavior space

Our **point of view** on organisation
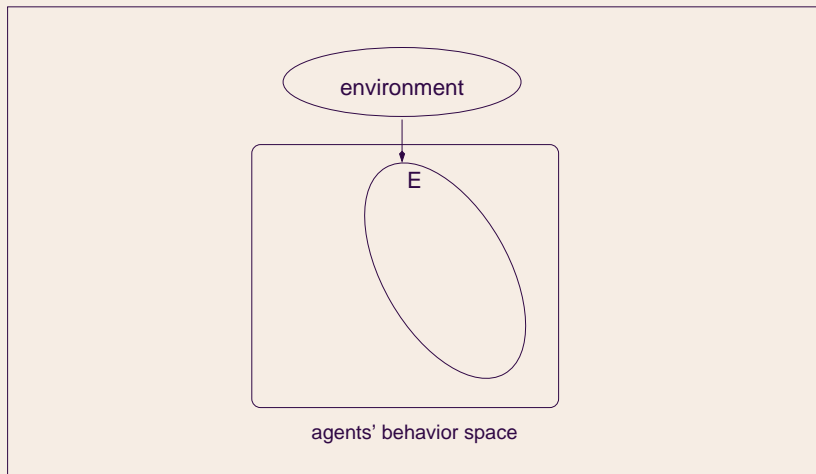
Our **point of view** on organisation

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

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

Our **point of view** on organisation
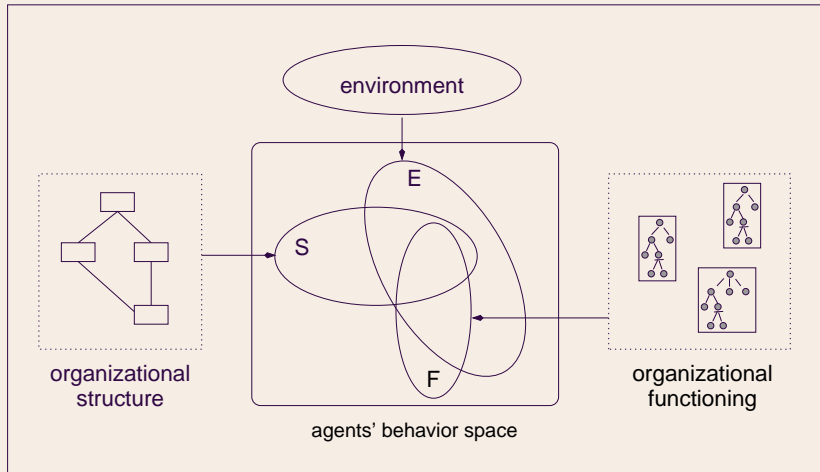
Goals, plans, missions, norms, ...
e.g.: TÆMS [Decker, 1998]

Our **point of view** on organisation
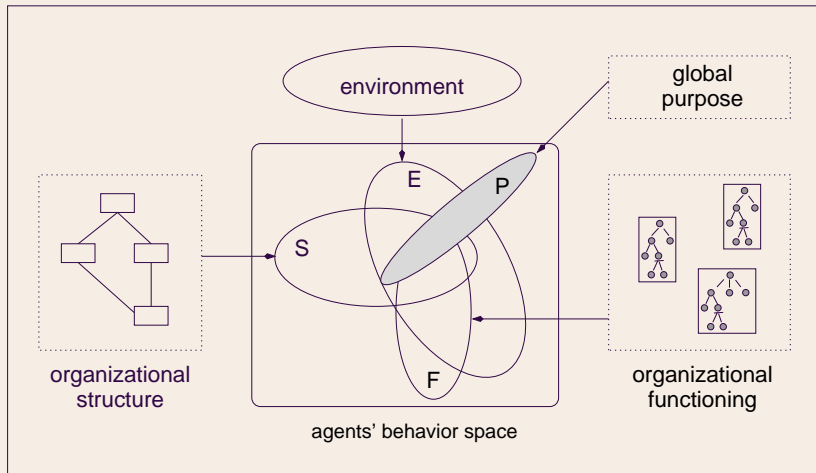
e.g.: TOVE , OPERA , STEAM

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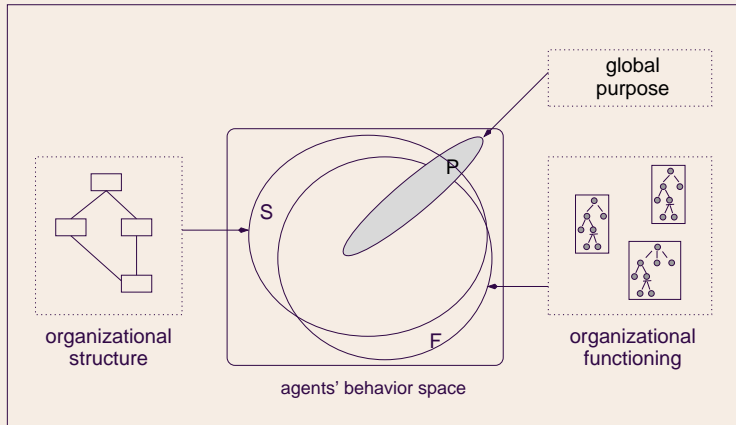
- The organisation does not lead to global purpose.

The **problem** of finding a good organisation II
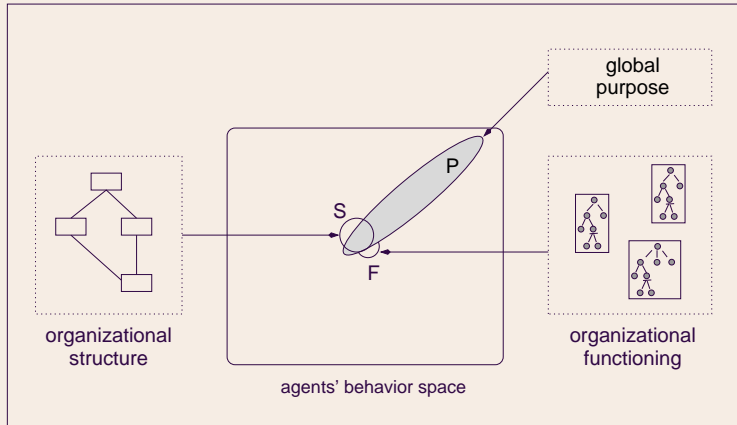
- The organisation extinguish the agents' autonomy.

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: $\mathcal{M}$OISE$^+$.

The $\mathcal{M}$OISE$^+$ organisational model

A proposal to join roles (structure) and plans (functioning) with some **independence** between them to simplify reorganisation. The $\mathcal{M}$OISE$^+$ is structured along three levels:

- *i*) Individual level: definition of the organisation's roles.
- *ii*) 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).
- *iii*) Collective level: the aggregation of roles in large structures.

Study Case: **Robocup** small size league I
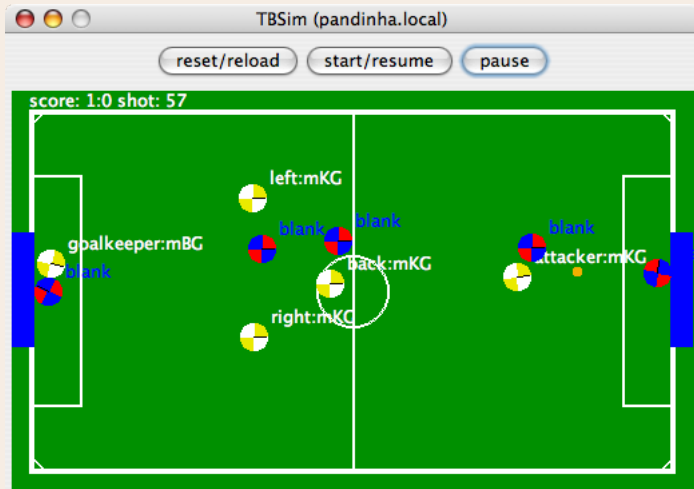
Study Case: **Robocup** small size league II
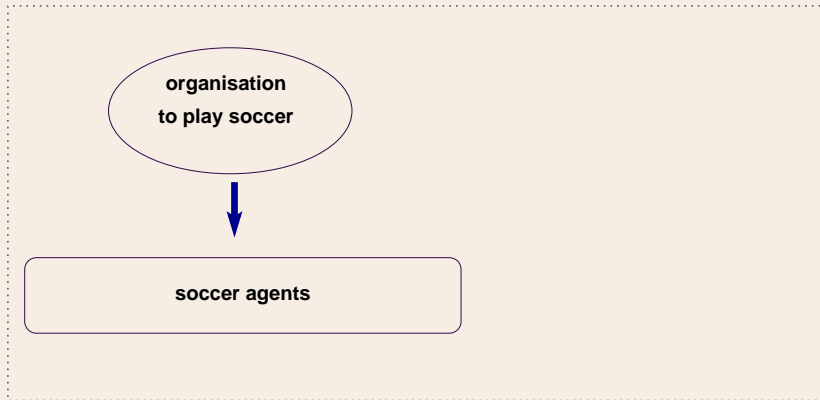
Specifying the JOJTEAM organisation: **structure** I
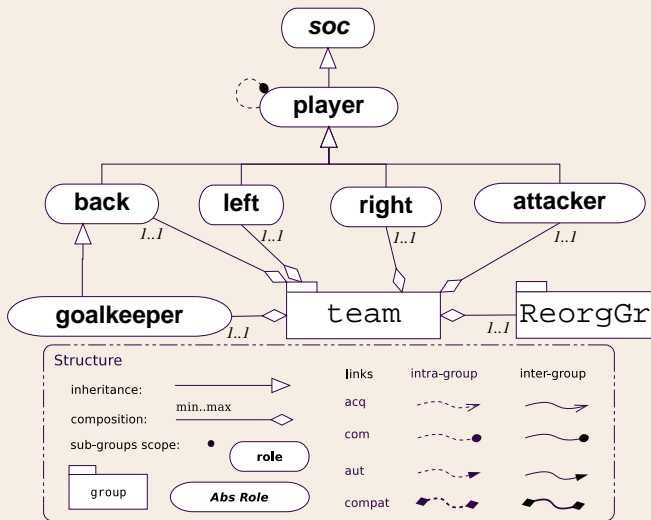
Specifying the JOJTEAM organisation: **structure** II
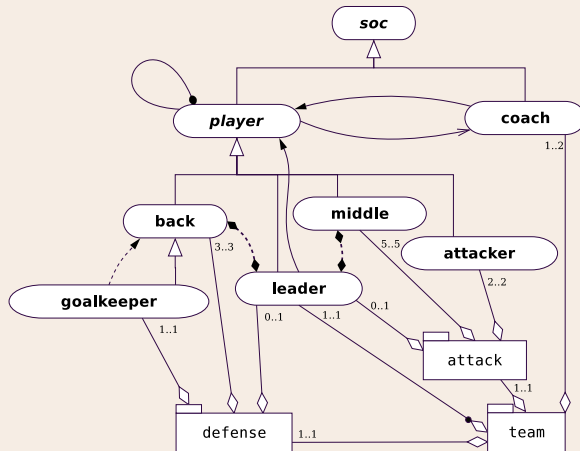
Specifying the JOJTEAM organisation: **functioning** I

	role	deontic	mission
mCG	back	*obligation*	*mKG*
	left	*obligation*	*mCG*
	right	*obligation*	*mCG*
	attacker	*obligation*	*mCG*
	goalkeeper	*obligation*	*mBG*

Specifying the JOJTEAM organisation: **functioning** II

m1, m2, m3
score a goal

m1
get the ball

m3
shot at the opponent's goal

m1
go towards the opponent field

m2
kick the ball to the goal area

m2
be placed in the middle field

m2
go to the opponent back line

m3
be placed in the opponent goal area

kick the ball to (agent committed to m2)
m1

Key

Scheme
missions **goal** *success rate*

sequence   choice   parallelism

Organizational Entity

Lucio ------- *m1*
Cafu -------- *m2*
Rivaldo ----- *m3*
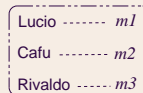
Approach to reorganise the team
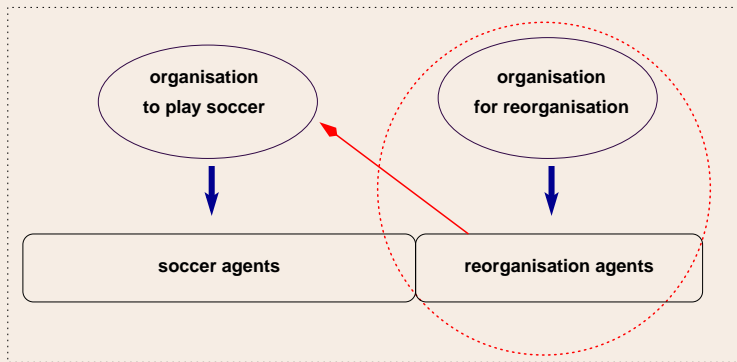
- *i)* Create a special group of agents specialised in reorganisation.
- *ii)* This new group is also organised.
- *iii)* Since the soccer agents follow the organisation, the new organisation is easily implemented.

Structural dimension of the **reorganisation**
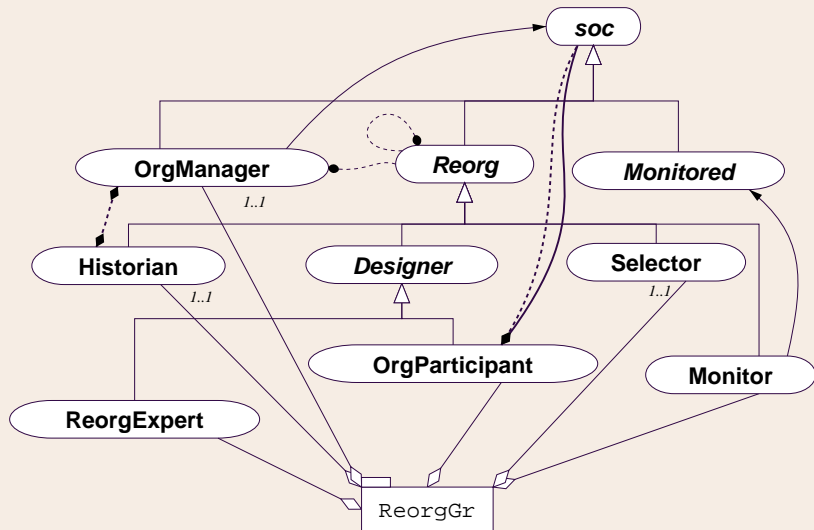
Functional dimension of the reorganisation

reorganization ^{m1}
$_{.8}$

monitoring ^{m2}

design(Fault) ^{m1}

implementation(Proposal) ^{m1}
$_{.9}$

selection(Proposals) ^{m6}

invitation ^{m1}

expertDes ^{m4} practiceDes ^{m5}
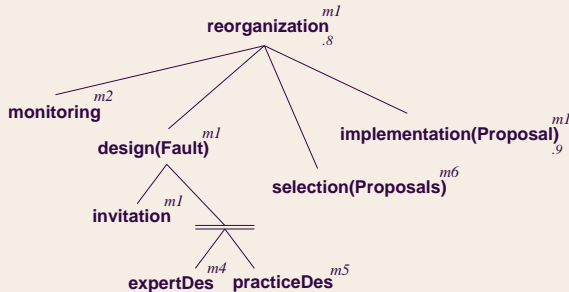
deontic relations:
$OrgManager \rightarrow obl(m_1)$
$Monitor \rightarrow obl(m_2)$
$ReorgExpert \rightarrow obl(m_4)$
$OrgParticipant \rightarrow per(m_5)$
$Selector \rightarrow obl(m_6)$

Example of **Monitoring** goal I

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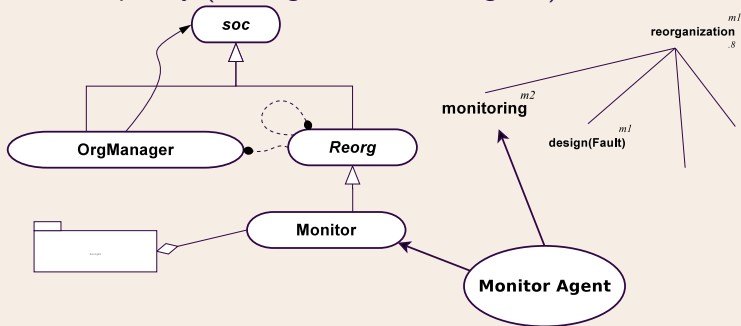
Design goal I

- JOJTEAM: 9 designers that always propose the same king of reorganisation ($1 \times 1 \times 3$, 4×1, increase the players area, change the team goals, ...)
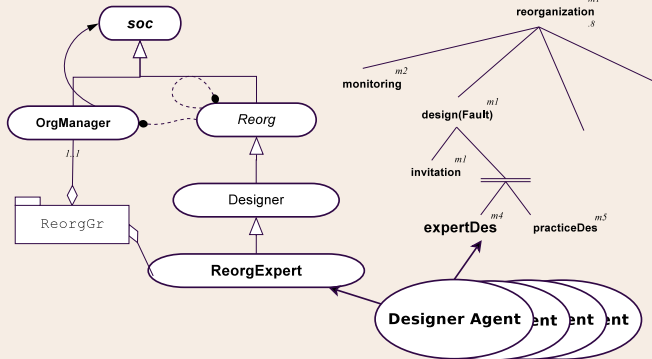
Design goal II

- 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

Implementation goal

- The OrgManager agent executes the reorganisation plan selected.

Results

Organisation
Reorganisation
Programming with (re)organisation
Requirements
S-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$
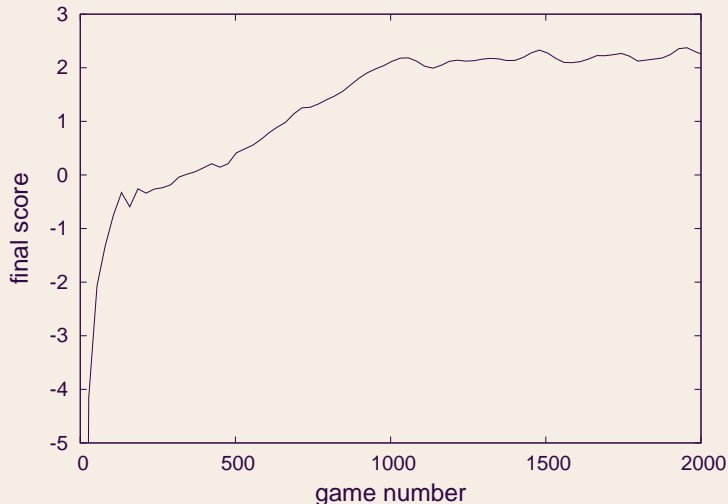
Programming organised agents

- How to implement MAS that follows 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!).
- System Centred approach:
  - Develop a multi-agent infrastructure that ensures that the organisational constraints will be followed.
  - The agents have to respect the organisation 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.

Organisation
Reorganisation
Programming with (re)organisation

Requirements
$\mathcal{S}$-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

$\mathcal{S}$-Moise^+: Saci + $\mathcal{M}\mathrm{oise}^+$
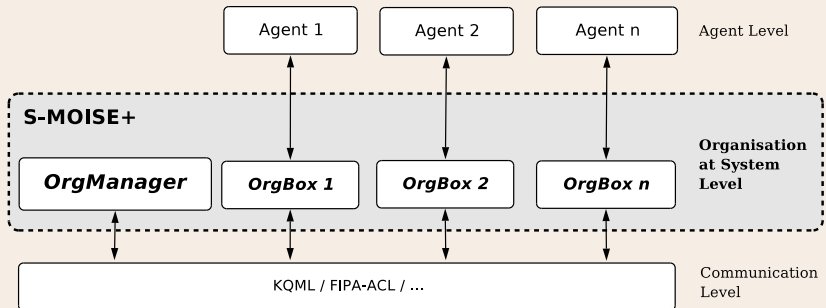
- Two mains components: OrgManager and OrgBox.

Organisation
Reorganisation
Programming with (re)organisation

Requirements
S-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

OrgBox – agent side

- The OrgBox is the interface that the agents use to access the organisational 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.

Organisation
Reorganisation
Programming with (re)organisation

Requirements
$\mathcal{S}$-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

OrgManager – service side

- 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.

Organisation
Reorganisation
Programming with (re)organisation

Requirements
$\mathcal{S}$-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

Organizational **Entity Dynamics**

The entity is changed by requests coming from agents' OrgBoxes.
Examples of messages' content:

- createGroup(team): create a new group based on the
  "team" group specification.

- createGroup(defence, g1): create a new $g1$ sub-group
  based on the "defence" specification.

- 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".

- ....

Organisation Requirements
Reorganisation $\mathcal{S}$-$\mathcal{M}$oise$^+$
Programming with (re)organisation $\mathcal{J}$-$\mathcal{M}$oise$^+$

Role adoption

The adoption of a role ρ by an agent α in the group g has the following constraints:

- The role ρ must belong to the specification of group g.

- The number of ρ players in g must be lesser or equals than the maximum number of ρ players defined in the specification of group g.

- For all roles ρ_i that agent α already plays in g, the roles ρ and ρ_i must be compatible in the specification of group g.

Organisation Requirements
Reorganisation $\mathcal{S}$-$\mathcal{M}$oise$^+$
Programming with (re)organisation $\mathcal{J}$-$\mathcal{M}$oise$^+$

Permitted goals and agent coordination for scheme execution

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

Organisation
Reorganisation
Programming with (re)organisation

Requirements
$\mathcal{S}$-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

Permitted goals and agent coordination for scheme execution

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

Organisation
Reorganisation
Programming with (re)organisation

Requirements
$\mathcal{S}$-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

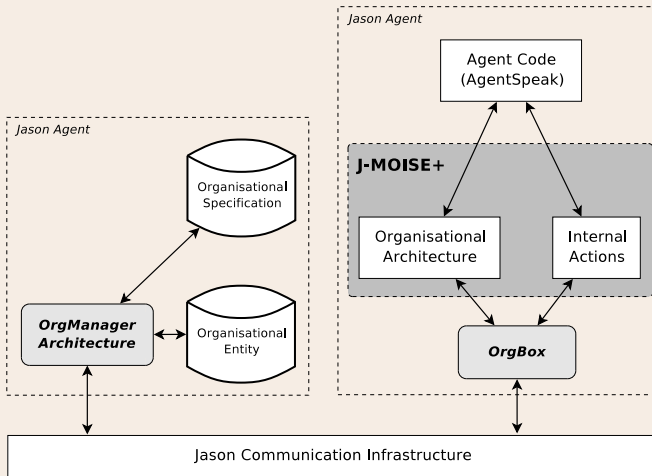# Permitted goals and agent coordination for scheme execution

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

Organisation
Reorganisation
Programming with (re)organisation

Requirements
$\mathcal{S}$-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

$\mathcal{J}$-Moise$^+$: **Jason**+ Moise$^+$

- $\mathcal{S}$-Moise$^+$ provides that organisational constraints are followed, but does not help us to program the agents or the agent reasoning about its organisation.
- $\mathcal{J}$-Moise$^+$
  - Programming agents with AgentSpeak.
  - BDI agents (reactive planning) – higher abstraction level.
  - Enable users 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.

Organisation
Reorganisation
Programming with (re)organisation

Requirements
S-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathbf{M}$oise$^+$

General view

Organisation
Reorganisation
Programming with (re)organisation

Requirements
S-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

Organisational Actions in AgentSpeak

- Example:

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

- Some available Organisational Actions:
  - `createGroup(<GrSpecId>[,<GrId>])`
  - `removeGroup(<GrId>)`
  - `createScheme(<SchSpecId>)`
  - `removeScheme(<SchId>)`
  - `adoptRole(<RoleId>,<GrId>)`
  - `removeRole(<RoleId>,<GrId>)`
  - `commitMission(<MisId>,<SchId>)`
  - `removeMission([<MisId>,] <SchId>)`

Organisation
Reorganisation
Programming with (re)organisation
Requirements
S-Moise[+]
J-Moise[+]

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(defence,Id) : true
    <- jmoise.adoptRole(back,Id).
  ```

  or

  ```
  +group(defence,Id)[owner(O)] : myFriend(O)
    <- jmoise.adoptRole(back,Id).
  ```

- Some group is destroyed

  ```
  -group(defence,Id) : true
    <- .print("The group ",Id," was removed!").
  ```

Organisation
Reorganisation
Programming with (re)organisation

Requirements
$\mathcal{S}$-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

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.

Organisation
Reorganisation
Programming with (re)organisation

Requirements
S-$\mathcal{M}$oise$^+$
$\mathcal{J}$-$\mathcal{M}$oise$^+$

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)
    <- ....
```

Summary I

The $\mathcal{M}OISE^+$ 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 $\mathcal{M}OISE^+$ 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 $\mathcal{M}OISE^+$ independence between struncture and functioning simplifies the construction of reorganisation plans.

Summary II

- $\mathcal{S}$-$\mathcal{M}$OISE$^+$:
  - 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

- $\mathcal{J}$-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

Further work

- Although implemented for $\mathcal{M}\text{OISE}^{+}$ 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 I

Bordini, R. H., Hübner, J. F., and Vieira, R. (2005).
Jason and the Golden Fleece of agent-oriented programming.
In Bordini, R. H., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., editors, *Multi-Agent Programming: Languages, Platforms, and Applications*, number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations, chapter 1. Springer.

de Almeida Júdice Gamito Dignum, M. V. F. (2003).
A model for organizational interaction: based on agents, founded in logic.
PhD thesis, Universiteit Utrecht.

Decker, K. S. (1998).
Task environment centered simulation.
In Prietula, M. J., Carley, K. M., and Gasser, L., editors, *Simulating Organizations: Computational Models of Institutions and Groups*, chapter 6, pages 105–128. AAAI Press / MIT Press, Menlo Park.

Esteva, M., Rodríguez-Aguilar, J. A., Rosell, B., and L., J. (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.

References II

Fox, M. S., Barbuceanu, M., Gruninger, M., and Lon, J. (1998).
An organizational ontology for enterprise modeling.
In Prietula, M. J., Carley, K. M., and Gasser, L., editors, *Simulating Organizations: Computational Models of Institutions and Groups*, chapter 7, pages 131–152. AAAI Press / MIT Press, Menlo Park.

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

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.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2004).
Using the $\mathcal{M}\textsc{oise}^+$ for a cooperative framework of MAS reorganisation.
In Bazzan, A. L. C. and Labidi, S., editors, *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence (SBIA'04)*, volume 3171 of *LNAI*, pages 506–515, Berlin. Springer.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2006).
$\mathcal{S}\text{-}\mathcal{M}\textsc{oise}^+$: A middleware for developing organised multi-agent systems.
In Boissier, O., Dignum, V., Matson, E., and Sichman, J. S., editors, *Proceedings of the International Workshop on Organizations in Multi-Agent Systems, from Organizations to Organization Oriented Programming in MAS (OOOP'2005)*, volume 3913 of *LNCS*. Springer.

References III

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.

Tambe, M., Pynadath, D. V., and Chauvat, N. (2001).
Building dynamic agent organizations in cyberspace.
IEEE Internet Computing, 4(2).

Full AgentSpeak code for a back player I

```
// when a defence group is created, adopt the role back in it
+group(defence,GId) : true
   <- jmoise.adoptRole(back,GId).

// when I see the ball, creates a sideAttack scheme
// with the team instance as responsible group
+see(ball)
   : group(team,GId) // get the id of the team group
   <- jmoise.createScheme(sideAttack, [GId]).

// when I has permission, commit to
// mission m1 of the scheme I have created
+permission(Sch, m1)
   : .myName(Me) &
     scheme(sideAttack,Sch)[owner(Me)]
   <- jmoise.commitMission(m1,Sch).
```

Full AgentSpeak code for a back player II

/* Plans for the organisational goals of mission m_1 */

```
+!getBall[scheme(Sch)] :  true
   <- <actions to bet the ball>;
      jmoise.setGoalState(Sch,getBall,satisfied).

+!goOpField[scheme(Sch)] :  true
   <- <actions to go to the opponent field>;
      jmoise.setGoalState(Sch,goOpField,satisfied).

+!kickBall[scheme(Sch)]
   :  // get the agent committed to m2
      commitment(Ag, m2, Sch)
   <- <actions to kick the ball to Ag>;
      jmoise.setGoalState(Sch,kickBall,satisfied).
```

Full AgentSpeak code for a back player III

```
// when the root goal of the scheme is satisfied,
// remove my missions
+goalState(Sch, G[root], satisfied) :  true
   <- jmoise.removeMission(Sch).

// removes the scheme if it has no more players
// and it was created by me
+schPlayers(Sch,0)
   :  .myName(Me) & scheme(_, Sch)[owner(Me)]
   <- jmoise.removeScheme(Sch).
```