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

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 organisations being an explicit description of the structure of the agents in the MAS in order to help them
To organisations as the declarative and explicit definition of the coordination scheme aiming at “controlling/coordinating” the global reasoning of the MAS

Norms

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 an user name
  - 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)
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 to clearly define the coordination of the MAS:
  - anchored/situated in the organisation
  - i.e. norms refer to organisational concepts (roles, groups, â€œ)
- Norms are interpreted and considered in the context of the organisation entity
- Organisation management mechanisms are complemented with norms management mechanisms (enforcement, regimentation, …)

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/permited/obliged behaviors constrained by the normative organisation

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

Using organisational concepts
To define a cooperative pattern
Programmed outside of the agents and outside of the environment
Program = Specification
By changing the organisation, we can change the MAS overall behaviour
First approach
- Agents read the program and follow it

Second approach
- Agents are forced to follow the program
- Agents are rewarded if they follow the program
- Agents are sanctioned in the other case
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+[Hübner et al., 2002], ...
- Specific anchors for situating organisations within the environment
  - e.g. embodied organisations [Piunti et al., 2009a]

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], MOISE+[Hübner et al., 2008], ORA4MAS [Hübner et al., 2009], ...
- Adaptation mechanisms, i.e. reorganisation infrastructure

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. MOISE+[Hübner et al., 2007], Autonomy based reasoning [Carabelea, 2007], ProsA[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. [de Brito et al., 2012], [Piunti et al., 2009b], [Okuyama et al., 2008]

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: 
**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: 
**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

An organisational specification is required to enable agents 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

Motivations for OOP: 
**Organisation** point of view

An organisational specification is required to enable the organisation 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
Origins and Fundamentals

Some OOP approaches
- AGR
- STEAM
- ISLANDER
- 2OPL
- OISE Framework
- Organisation Modeling Language (OML)
- Organisation Management Infrastructure (OMI)
- Org. Embodiement Mechanisms for Cartago (E-O)
- Org. Awareness Mechanisms in Jason (A-O)

Summary

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
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 [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 OML Modelling Dimensions

B: agents’ possible behaviors
P: agents’ behaviors that lead to global purpose
E: agents’ possible behaviors constrained by the environment
$O_i$: agents’ possible behaviors structurally constrained by the organization
$O_i'$: agents’ possible behaviors functionally constrained by the organization
STEAM OMI: TEAMCORE [Pynadath and Tambe, 2003]

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

<table>
<thead>
<tr>
<th>B: agents’ possible behaviors</th>
<th>P: agents’ behaviors that lead to global purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>O&lt;sub&gt;S&lt;/sub&gt;: agents’ possible behaviors constrained by the environment</td>
<td>E: agents’ possible behaviors interactionally constrained by the organisation</td>
</tr>
</tbody>
</table>

ISLANDER OML: IDL [Esteva et al., 2001]
The aim is to design and develop a programming language to support the implementation of coordination mechanisms in terms of normative concepts.

An organisation:
- determines effect of external actions
- normatively assesses effect of agents’ actions (monitoring)
- sanctions agents’ wrongdoings (enforcement)
- prevents ending up in really bad states (regimentation)

**Example (Train Station)**

Facts:
\[
\{ -at\_platform, -in\_train, -ticket \}
\]

Effects:
\[
\begin{align*}
&{ -at\_platform } \rightarrow { at\_platform }, \\
&{ -ticket } \rightarrow { ticket }, \\
&{ at\_platform, -in\_train } \rightarrow \text{embark}
\end{align*}
\]

Counts as rules:
\[
\begin{align*}
& { at\_platform, -ticket } \rightarrow \{ \text{viol\_ticket} \}, \\
& { in\_train, -ticket } \rightarrow \{ \text{viol\_i} \}
\end{align*}
\]

Sanction rules:
\[
\begin{align*}
& \{ \text{viol\_ticket} \} \rightarrow \{ \text{fined\_10} \}
\end{align*}
\]
Several models
Several dimensions on modelling organisation
- Structural (roles, groups, ...)
- Functional (global plans, ....)
- Dialogical (scenes, protocols, ...)
- Normative (norms)
OMISE Modelling Dimensions

Structural Specification
- Groups, links, roles
- Compatibilities, multiplicities
- Inheritance

Functional Specification
- Global goals, plans, Missions, schemas, preferences

Normative Specification
- Permissions, Obligations
- Allows agents autonomy!

OMISE OML

- OML for defining organisation specification and organisation entity
- Three independent dimensions [Hübner et al., 2007]
  - Structural: Roles, Groups
  - Functional: Goals, Missions, Schemes
  - Normative: Norms (obligations, permissions, interdictions)
- Abstract description of the organisation for
  - the designers
  - the agents
  - ORA4MAS [Hübner et al., 2009]
  - the Organisation Management Infrastructure
- Three independent dimensions [Hübner et al., 2007]
  - Structural: Roles, Groups
  - Functional: Goals, Missions, Schemes
  - Normative: Norms (obligations, permissions, interdictions)

OMISE OML meta-model (partial & simplified view)

- Cardinalities are not represented
- Composition, association, concept mapping
- Structural spec., functional spec., normative spec.
### 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

### 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 (subgroups tag)
  - formation constraints on the components of the group (formation-constraints)

Example

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

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
  - a scope (scope)
  - and validity to subgroups (extends-subgroups)

Example

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

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-subgroups and directions (bi-dir)

Example

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

**Example**

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

Goal States from the Organization Point of View

- waiting: initial state
- enabled: 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

Note: goal state from the Organization point of view may be different of the goal state from the Agent point of view

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

```xml
<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

Normative Specification

- Explicit relation between the functional and structural specifications
- Permissions and obligations to commit to missions in the context of a role
- The normative specification 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

```xml
<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

<table>
<thead>
<tr>
<th>role</th>
<th>deontic</th>
<th>mission</th>
<th>TTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>back</td>
<td>obliged</td>
<td>m1 get the ball, go...</td>
<td>1 minute</td>
</tr>
<tr>
<td>left</td>
<td>obliged</td>
<td>m2 be placed at ..., kick...</td>
<td>3 minute</td>
</tr>
<tr>
<td>right</td>
<td>obliged</td>
<td>m2 kick to the goal, ...</td>
<td>1 day</td>
</tr>
<tr>
<td>attacker</td>
<td>obliged</td>
<td>m3 kick to the goal, ...</td>
<td>30 seconds</td>
</tr>
</tbody>
</table>

<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"/>

...
Organisation management infrastructure (OMI)

**Responsibility**
- Managing – coordination, regulation – the agents’ execution within organisation defined by an organisational specification

Organisation

Program

OMI

(e.g. MadKit, AMELI, OISE, ...)
Obligations life cycle

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

Example (role cardinality norm – regimentation)

$\text{group\_role}(\text{writer},1,5)$.

$\text{norm\ ncar}: \text{group\_role}(R,_,M) \land \text{rplayers}(R,G,V) \land V > M$ $\implies \text{fail}(\text{role\_cardinality}(R,G,V,M))$.

Example (role cardinality norm – agent decision)

$\text{norm\ ncar}: \text{group\_role}(R,_,M) \land \text{rplayers}(R,G,V) \land V > M \land \text{plays}(E,\text{editor},G)$ $\implies \text{oimg}(E,\text{ncar,committed}(E,ms,\_),\text{\{now + 1 hour\}})$.

ORA4MAS– GroupBoard artifact

Manages the functioning of an instance of group in the organization.

- **Operations**:
  - $\text{adoptRole}(\text{role})$ (resp. $\text{leaveRole}(\text{role})$): attempts to adopt (resp. leave) role in the group
  - $\text{addScheme}(\text{schid})$ (resp. $\text{removeScheme}(\text{schid})$): attempts to set (resp. unset) the group responsible for the scheme managed by the SchemeBoard

- **Observable Properties**:
  - $\text{specification}$: group spec. in the OS
  - $\text{player}$: list of players of role in the group
  - $\text{schemes}$: list of scheme identifiers that the group is responsible for

ORA4MAS– SchemeBoard artifact

Manages the functioning of an instance of social scheme in the organization.

- **Operations**:
  - $\text{commitMission}(\text{mission})$ (resp. $\text{leaveMission}$): attempts to "commit" (resp. "leave") a mission in the scheme
  - $\text{goalAchieved}$(goal): declares that goal is achieved
  - $\text{setArgumentValue}(\text{goal, argument, value})$: defines the value of goal’s argument

- **Observable Properties**:
  - $\text{specification}$: scheme spec. in the OS
  - $\text{commitments}$: list of commitments to missions in the scheme
  - $\text{groups}$: list of groups resp. for the scheme
  - $\text{goalState}$: list of goals’ current state
  - $\text{obligation}$: list of active obligations in the scheme
Constitutive rules

### Count-As rule

An event occurring on an artifact, in a particular context, may transform the events created in the working environment into activation of an organisational operation.

### 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, ...)

Environment integration

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

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.

**Example (GroupBoard)**

```plaintext
joinWorkspace("ora4mas",O4MWsp);
makeArtifact(
   "auction",
   "ora4mas.nopl.GroupBoard",
   ["auction-os.xml", auctionGroup, false, true ],
   GrArtId);
adoptRole(auctioneer);
focus(GrArtId);
```
Organisational actions in Jason II

- For groups:
  - create_group
  - remove_group

Example

```java
... .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 III

Example (SchemeBoard)

```java
... makeArtifact(
    "sch1",
    "ora4mas.nopl.SchemeBoard",
    ["auction-os.xml", doAuction, false, true ],
    SchArtId);
focus(SchArtId);
addScheme(Sch);
commitMission(mAuctioneer) [artifact_id(SchArtId)];
... 
```

Organisational actions in Jason IV

- For schemes:
  - create_scheme
  - add_responsible_group
  - remove_scheme
  - goal_achieved

Example

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

Organisational actions in Jason V

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

- `specification`
- `scheme_specification`
- `play(agent, role, group)`
- `commitment(agent, mission, scheme)`
- `goalState(scheme, goal, list of committed agents, list of agents that achieved the goal, state of the goal)`
- `obligation(agent,norm,goal,dead line)`
- `normFailure(norm)`

### 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)**

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

**Example (signals)**

```
+normFailure(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!").
```

### Inspection of agent bob (cycle #0)

**Beliefs**

```
commitment(bob,mManager,"sch2")
(artifact_id(coobj_4),
  cept),artifact_name(coobj_4,"sch2"),artifact_type(coobj_4,"ora4m
  commitment(bob,mManager,"sch1")
(artifact_id(coobj_3),
  cept),artifact_name(coobj_3,"sch1"),artifact_type(coobj_3,"ora4m
  current_wsp(coobj_1,"ora4mas"),308b05b0-2994-4f8
  formationStatus(ok)[artifact_id(coobj_2),obs_prop_id("obs_i
  obj_2"),mypaper),artifact_type(coobj_2,"ora4mas.npl.Group8n
  goalState("sch2",wp.[bob],[bob],satisfied)[artifact_id(co
```

---

**Actions**

**Example**

```
Actions
```

**Events**

**Example**

```
Events
```

**Summary**

**Example**

```
Summary
```

---

**Fundamentals**

**OOP**

**OML**

**OMI**

**E-O**

**A-O**

---

**May 2014**

**MAOP**

**Page 89 / 112**
<organisational-specification>
  <structural-specification>
    <role-definitions>
      <role id="author" />
      <role id="writer" extends role="author"/>
      <role id="editor" extends role="author"/>
    </role-definitions>

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

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)

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)
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

jaime (only wtitle is possible, Jaime should work)

event: +!wtitle

action: jmoise.set_goal_state(s1, wtitle, satisfied)

May 2014

Writing paper sample V

Execution

jaime event: +!wabs

action: jmoise.set_goal_state(s1, wabs, satisfied)

May 2014

Writing paper sample VI

Execution

jaime event: +!wsectitles

action: jmoise.set_goal_state(s1, wsectitles, satisfied)

May 2014

Writing paper sample VII

Execution

olivier, jomi event: +!wsecs

action: jmoise.set_goal_state(s1, wsecs, satisfied)

May 2014
May 2014

Writing paper sample VIII

execution

jaime event: +!wcon; ...

olivier event: +!wref; ...

jaime event: jmoise.remove_mission(s1)

olivier event: jmoise.remove_scheme(s1)

Useful tools — Mind inspector

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.sourceforge.net
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

- OISE 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

Bibliography I

  La sociologie des organisations.
  Seuil, 3ème edition.

  Introduction to the special issue on normative multiagent systems.

  Reasoning about autonomy in open multi-agent systems - an approach based on the social power theory.
  in french, ENS Mines Saint-Etienne.

  Programming institutional facts in multi-agent systems.
  In COIN-12, Proceedings.

  On the formal specification of electronic institutions.

Bibliography II

  AMELI: An agent-based middleware for electronic institutions.

  A meta-model for the analysis and design of organizations in multi-agents systems.

  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 III

  Moiseinst: An organizational model for specifying rights and duties of autonomous agents.

  The MadKit agent platform architecture.

  MOISE: An organizational model for multi-agent systems.
A normative organisation programming language for organisation management infrastructures.

Instrumenting Multi-Agent Organisations with Organisational Artifacts and Agents.
Journal of Autonomous Agents and Multi-Agent Systems.

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

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.

Co-ordination in Artificial Agent Societies: Social Structures and Its Implications for Autonomous Problem-Solving Agents, volume 1535 of LNAI. Springer.

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.

Embodied organisations in mas environments.