

Multi-Agent Oriented Programming

Organization Dimension



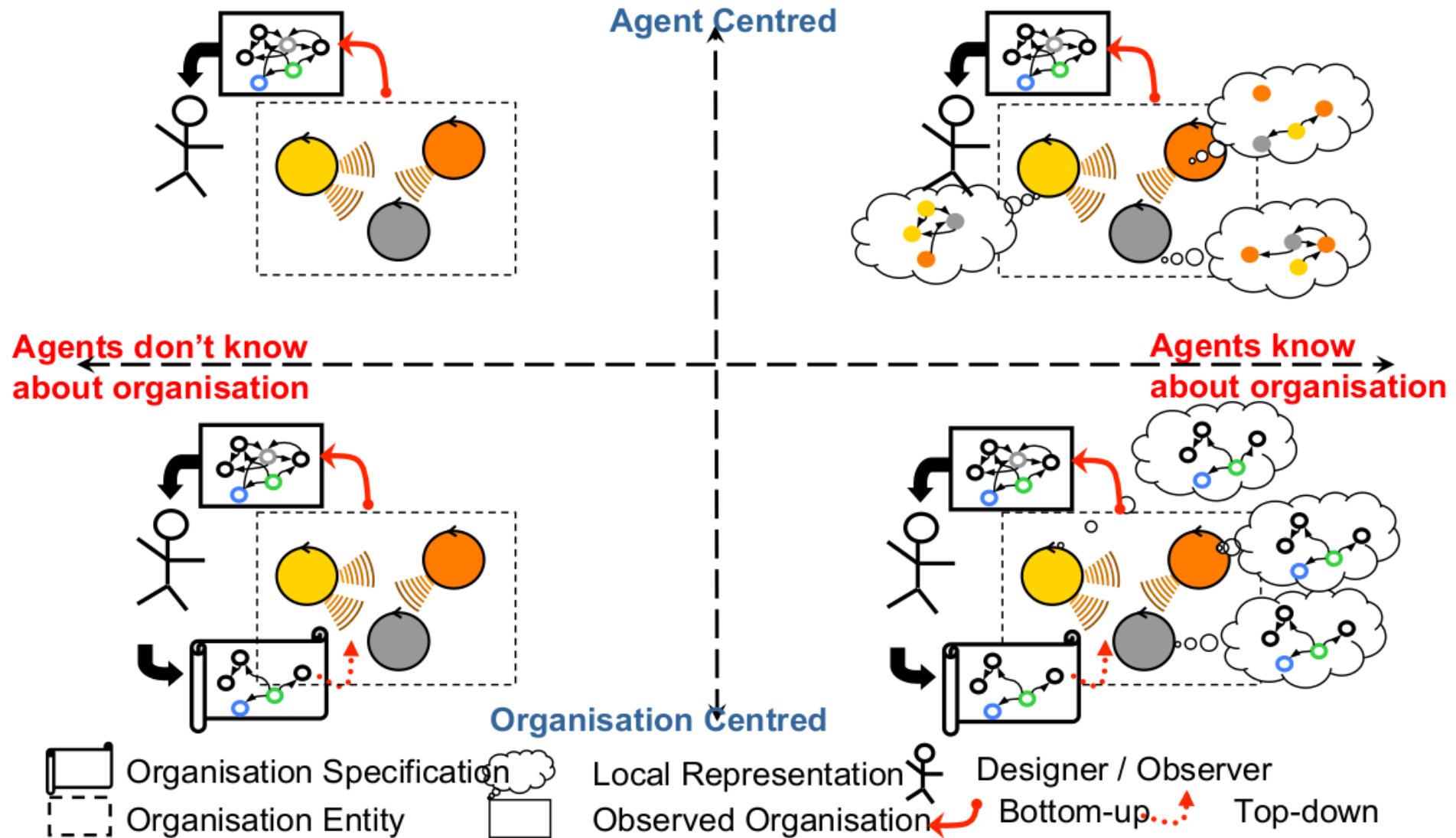
Introduction: Some definitions

- Organizations are structured, patterned systems of activity, knowledge, culture, memory, history, and capabilities that are distinct from any single agent (Gasser, 2001)
 - ~w> organizations 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)
 - ~w> definition by the designer, or by actors, to achieve a **purpose**
- An organization is characterized by: a division of tasks, a distribution of roles, authority systems, communication systems, contribution-retribution systems (Bernoux, 1985)
 - ~w> **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)
 - ~w> **pattern of emergent cooperation**

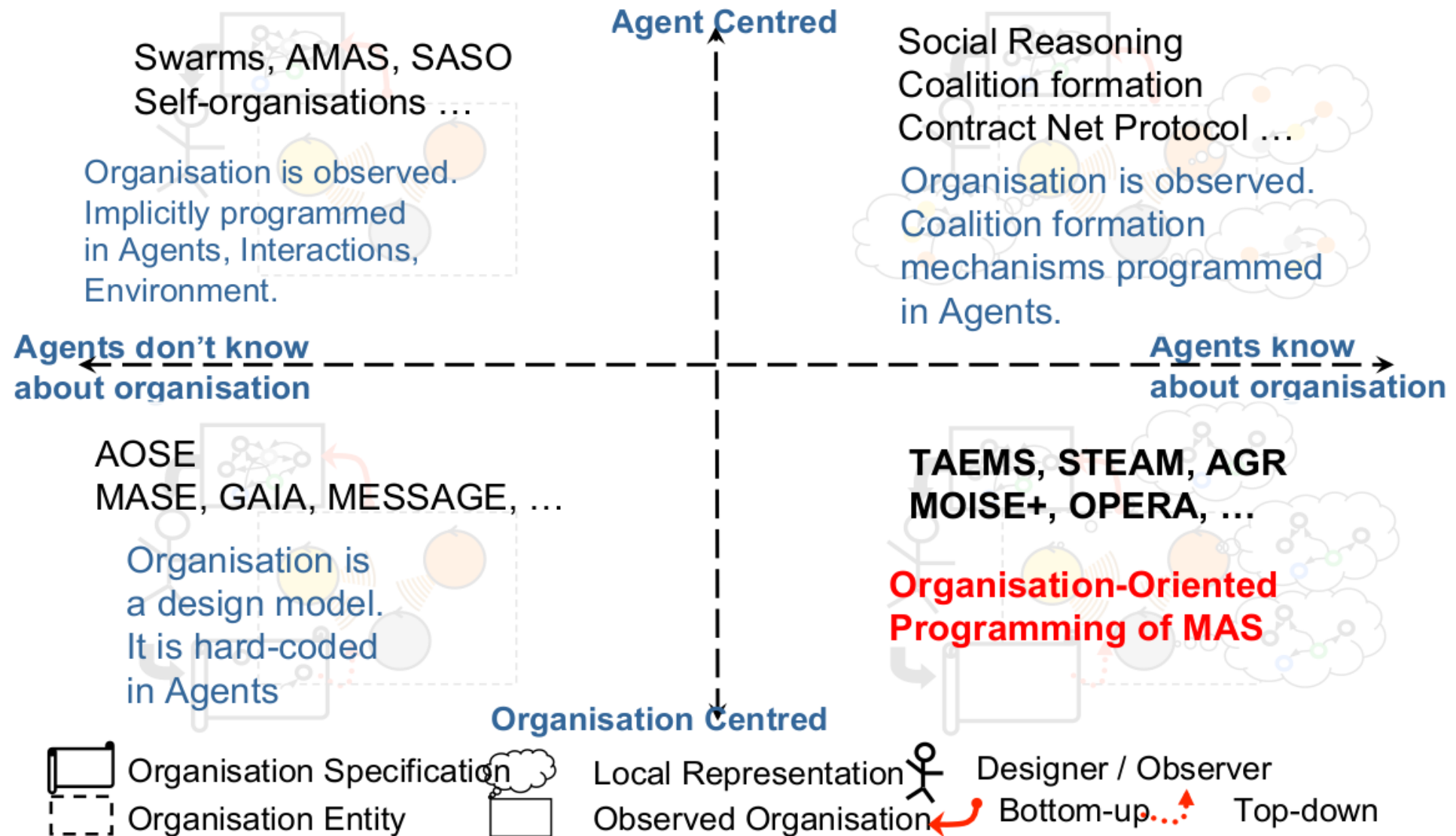
Organization in MAS

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

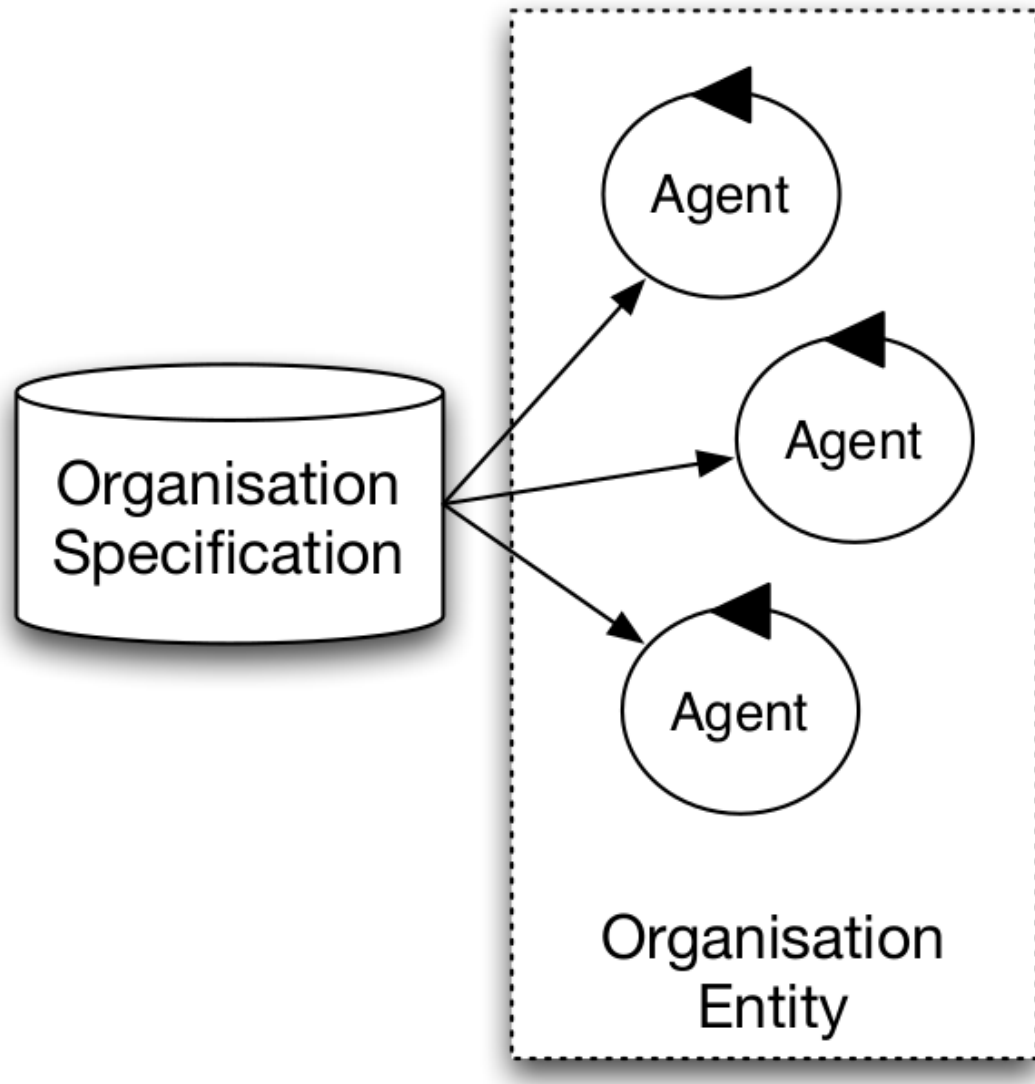
Perspective on Organizations



Perspective on Organizations



Organization Oriented Programming

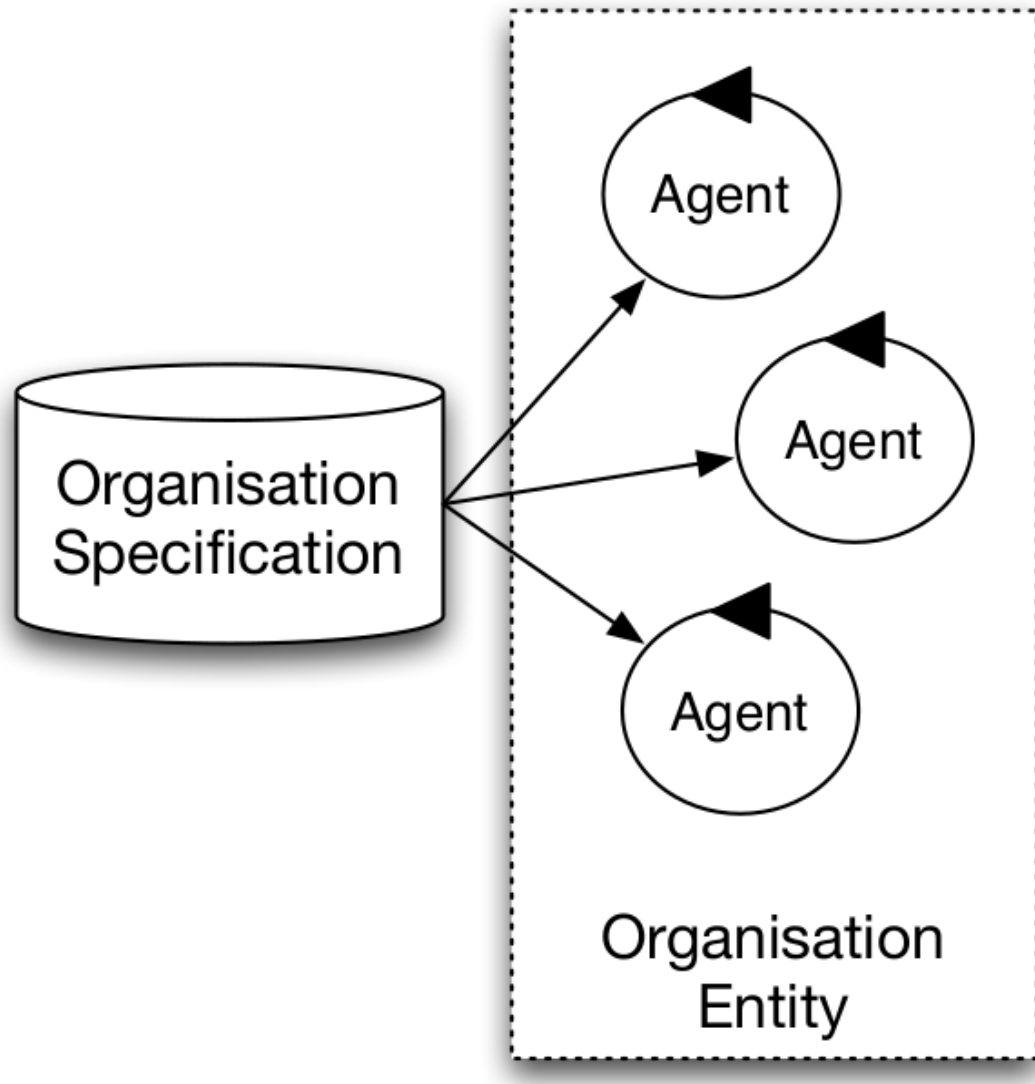


Programming **outside the agents** using of **organizational concepts** to **coordinating and regulating** autonomous agents

`Program = Specification`

By changing the specification, we can change the MAS behavior

Organization Oriented Programming



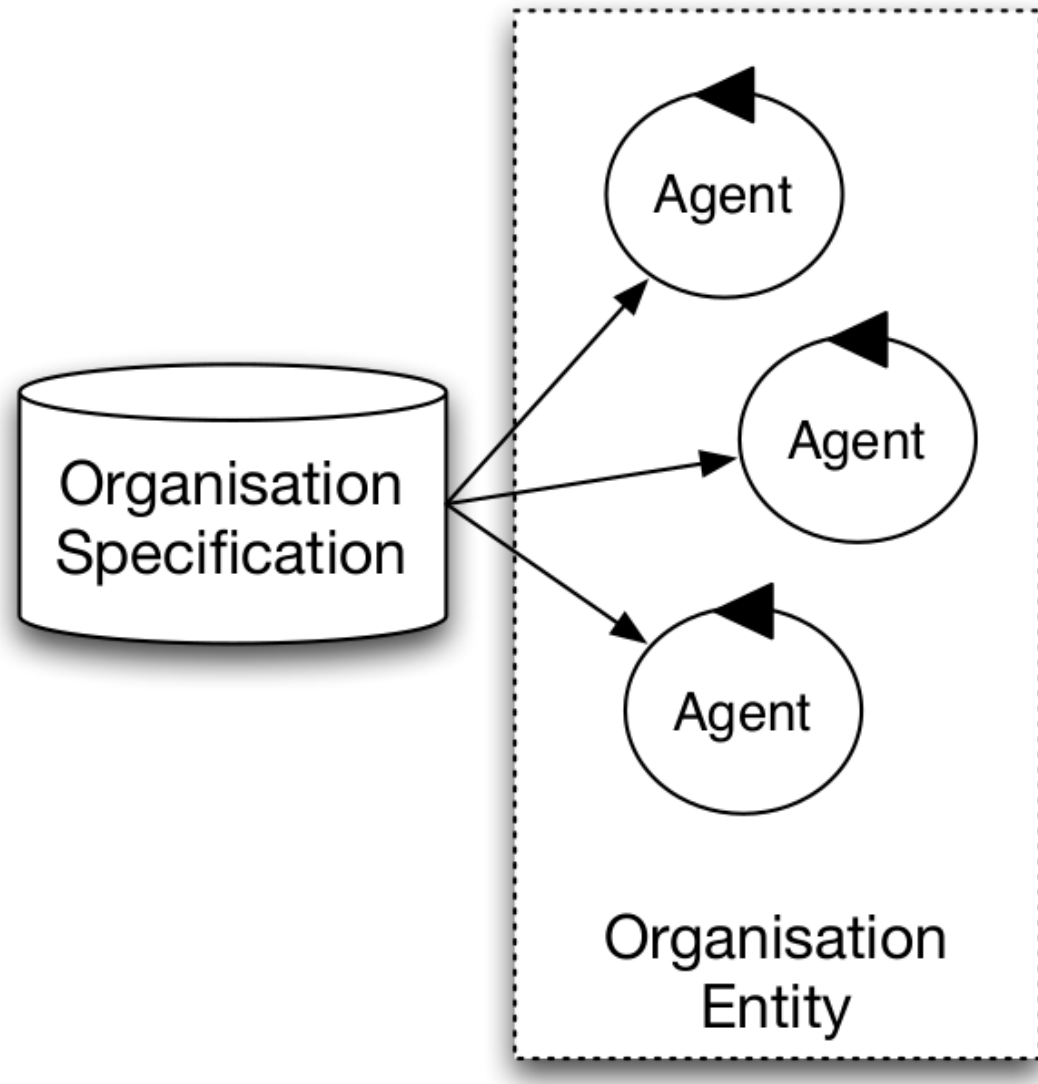
First approach

- Agents read the program/specification and follow it

Second approach

- Agents **are forced** to follow the program/specification
- Agents **are rewarded/punished** if they follow/not follow the program/specification

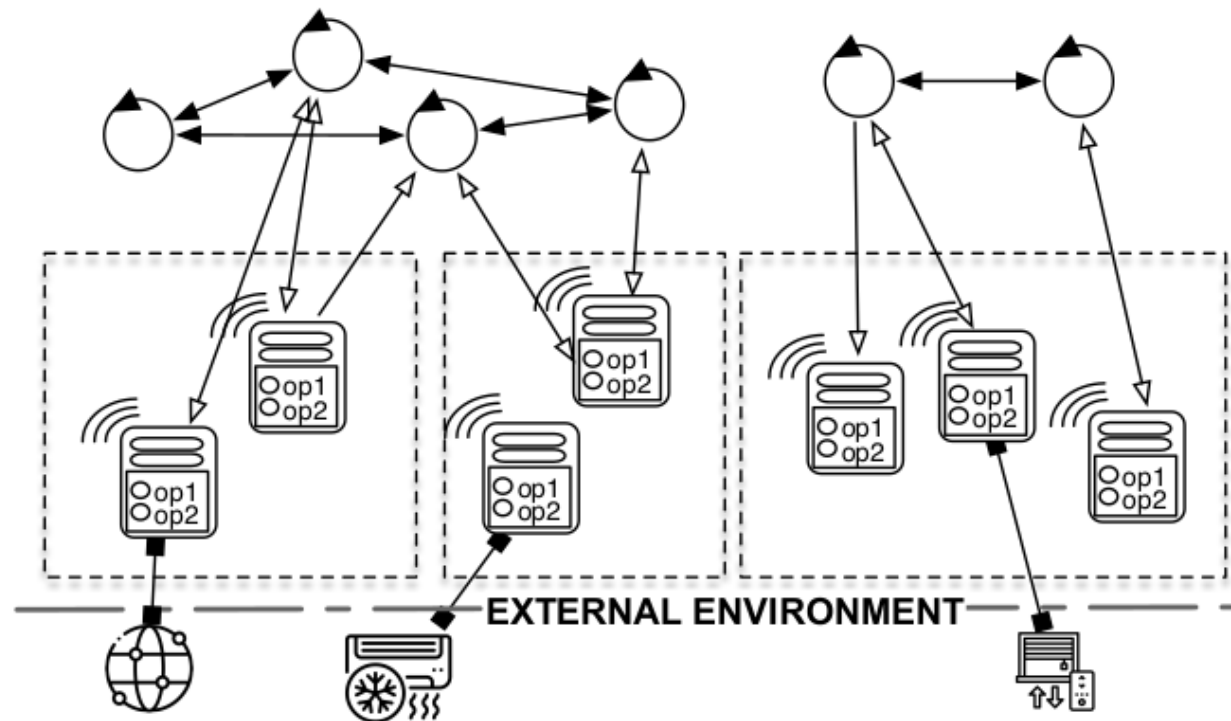
Organization Oriented Programming



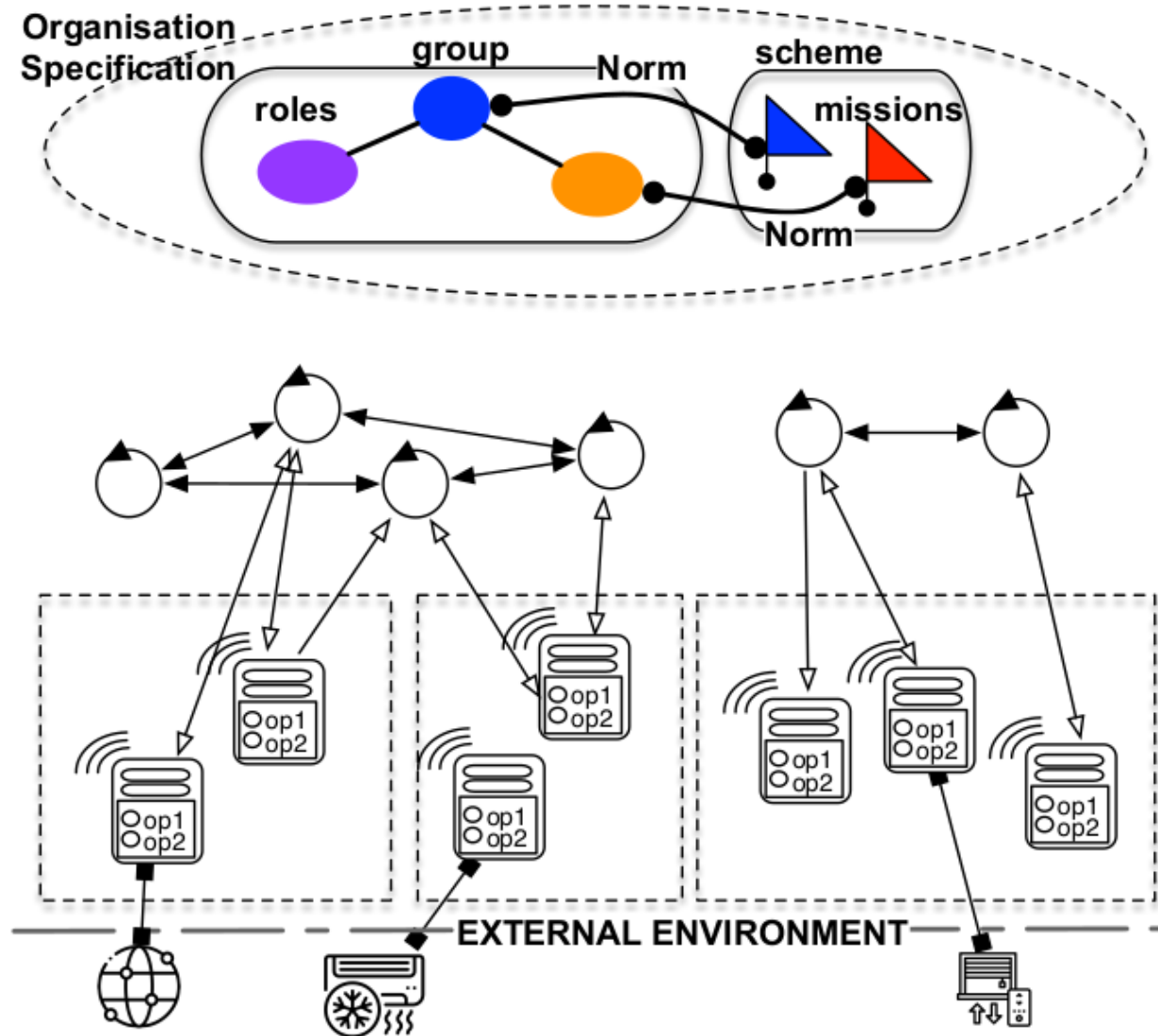
Components

- Programming language
- Platform
- Integration to agent architectures and to the environment

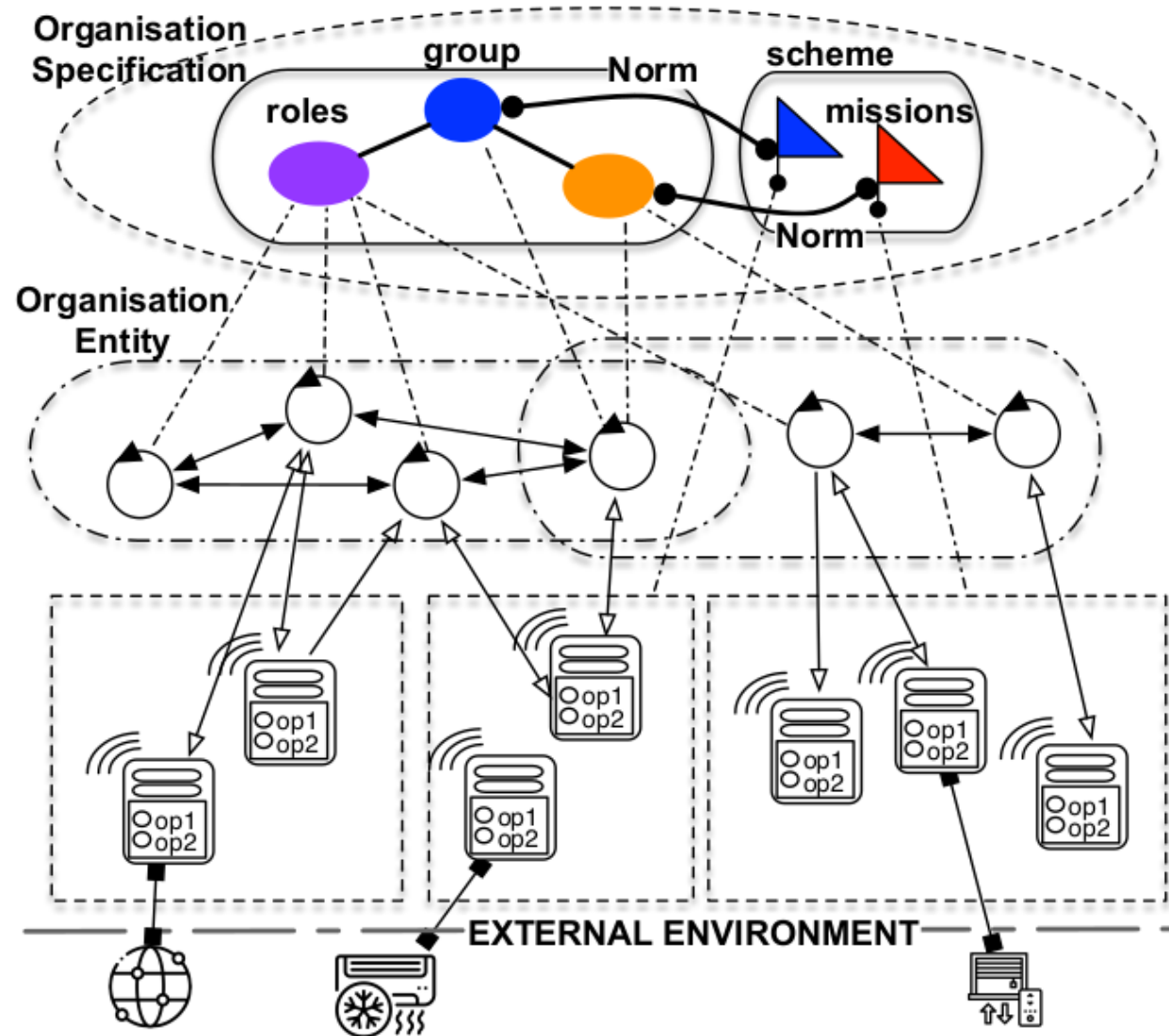
JaCaMo Organization Dimension



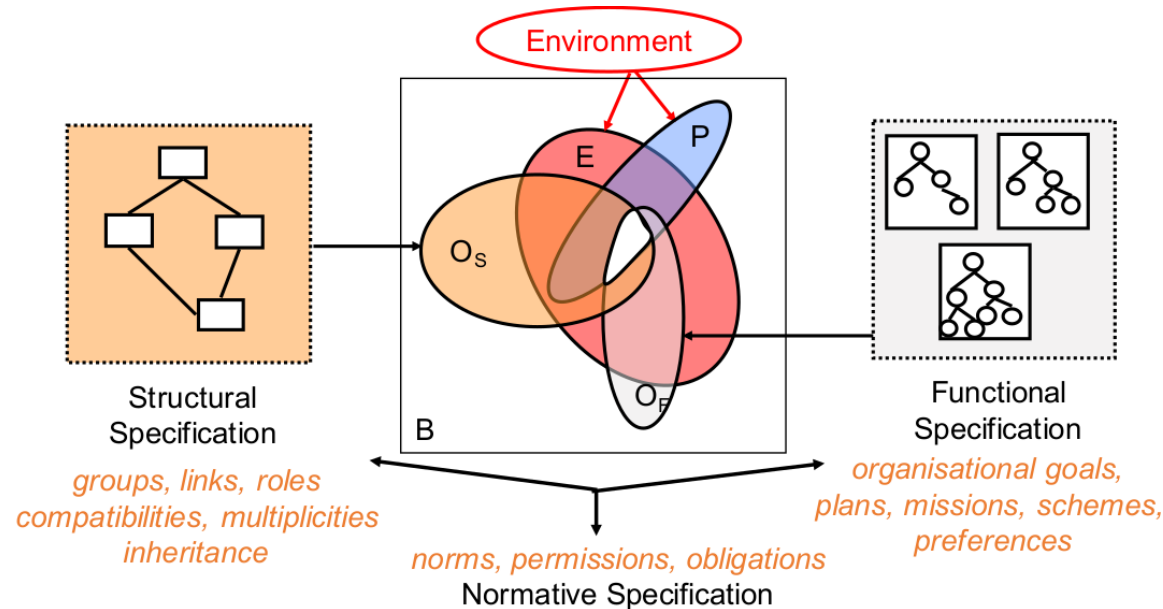
JaCaMo Organization Dimension



JaCaMo Organization Dimension



JaCaMo Organization Dimension



- Dimensions (Hübner et al. 2007)
 - **Structural** (i.e., Roles, Groups),
 - **Functional** (i.e., Organizational Goals, Missions, Schemes)
 - **Normative** (i.e., Norms with obligations, permissions, interdictions)
- Enable agent's autonomy w.r.t. organization (enforcement vs regimentation)
- Programming and representing the organization
 - make it accessible to the designers, the agents, the coordination and regulation management infrastructure (Hübner et al., 2010)

Basic Concepts

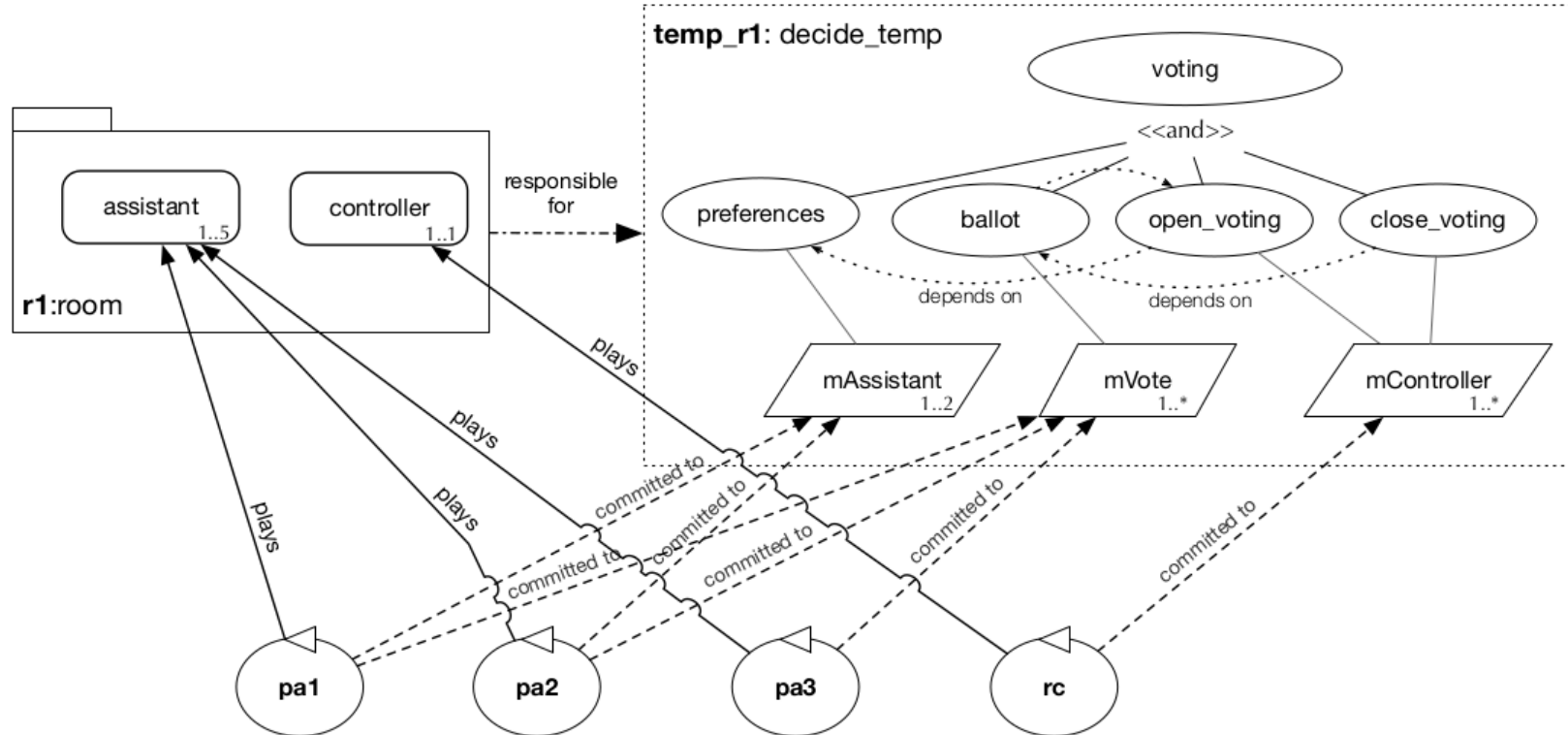
- **Organization:** abstractions to declare and make accessible to agents their (current and/or expected) collective coordinated and regulated relations and activities in the shared environment organization entity, organization specification

- **Group:** social context in which agents can play roles, undertake their expected coordinated behavior as well as their rights and duties.
- **Role:** statement that determines the interactions, relations, rights and duties taking place for an agent within a group.

- **Organizational goal:** state of affair that has to be (or has been) satisfied by one or several agents
- **Mission:** set of organizational goals that have to be achieved under the responsibility of an individual agent in the organization
- **Social scheme:** a goal decomposition tree executed in a coordinated manner under the responsibility of agents participating to a group in accordance to their rights and duties

- **Norm:** statement of the rights and duties of agents in the context of an organization

Declarative Organization Programming

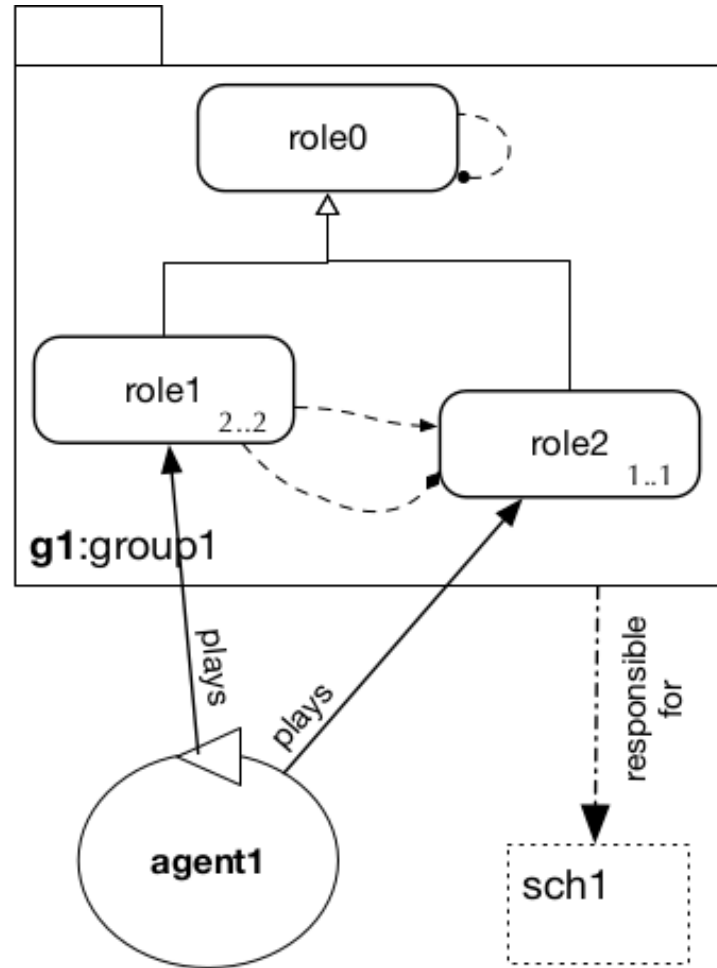


- Structural patterns (groups (r1:room), roles (assistant, controller), links)
- Coordination patterns (
 - goal decomposition trees (voting, preferences, ballot, open_voting, close_voting)
 - missions (mAssistant, mVote, mController))
- Rights and duties (norms)

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 rights and constraints on the behavior of agents playing it
 - **Link**: relation between roles that directly constrains the agents in their interaction with the other agents playing the corresponding roles
 - **Group**: set of links, roles, compatibility relations used to define a shared context for agents playing roles in it

Structural Specification

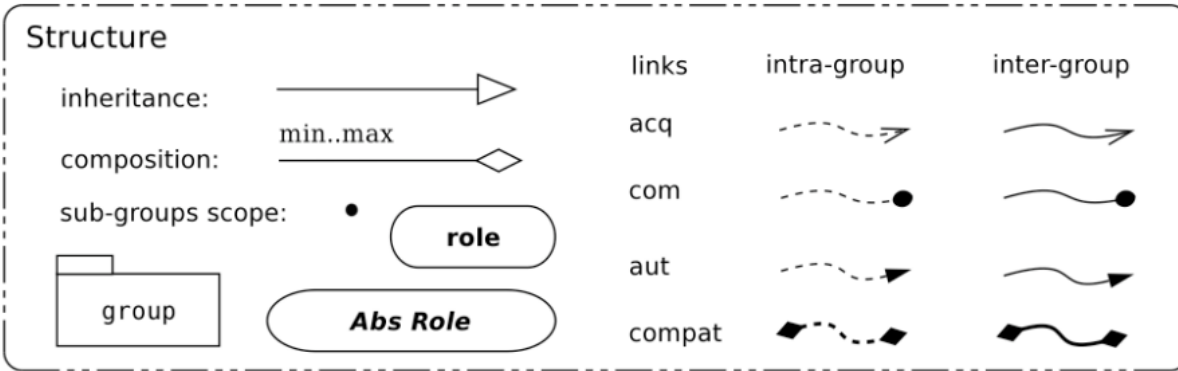
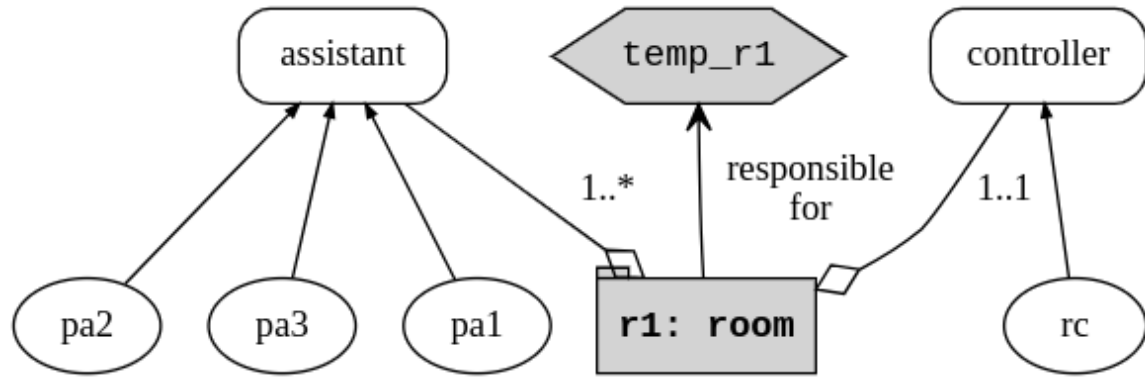


```

1  <structural-specification>
2
3  <role-definitions>
4      <role id="role0"/>
5      <role id="role1"> <extends role="role0"/> </role>
6      <role id="role2"> <extends role="role0"/> </role>
7  </role-definitions>
8
9  <group-specification id="group1">
10     <roles>
11         <role id="role1" min="1" max="2"/>
12         <role id="role2" min="1" max="1"/>
13     </roles>
14
15     <links>
16         <link from="role1" to="role2" type="authority"
17             scope="intra-group" bi-dir="false" />
18         <link from="role0" to="role0" type="communication"
19             scope="intra-group" bi-dir="true" />
20     </links>
21     <formation-constraints>
22         <compatibility from="role1" to="role2" bi-dir="true"/>
23     </formation-constraints>
24 </group-specification>
25 </structural-specification>

```

Structural Specification Example



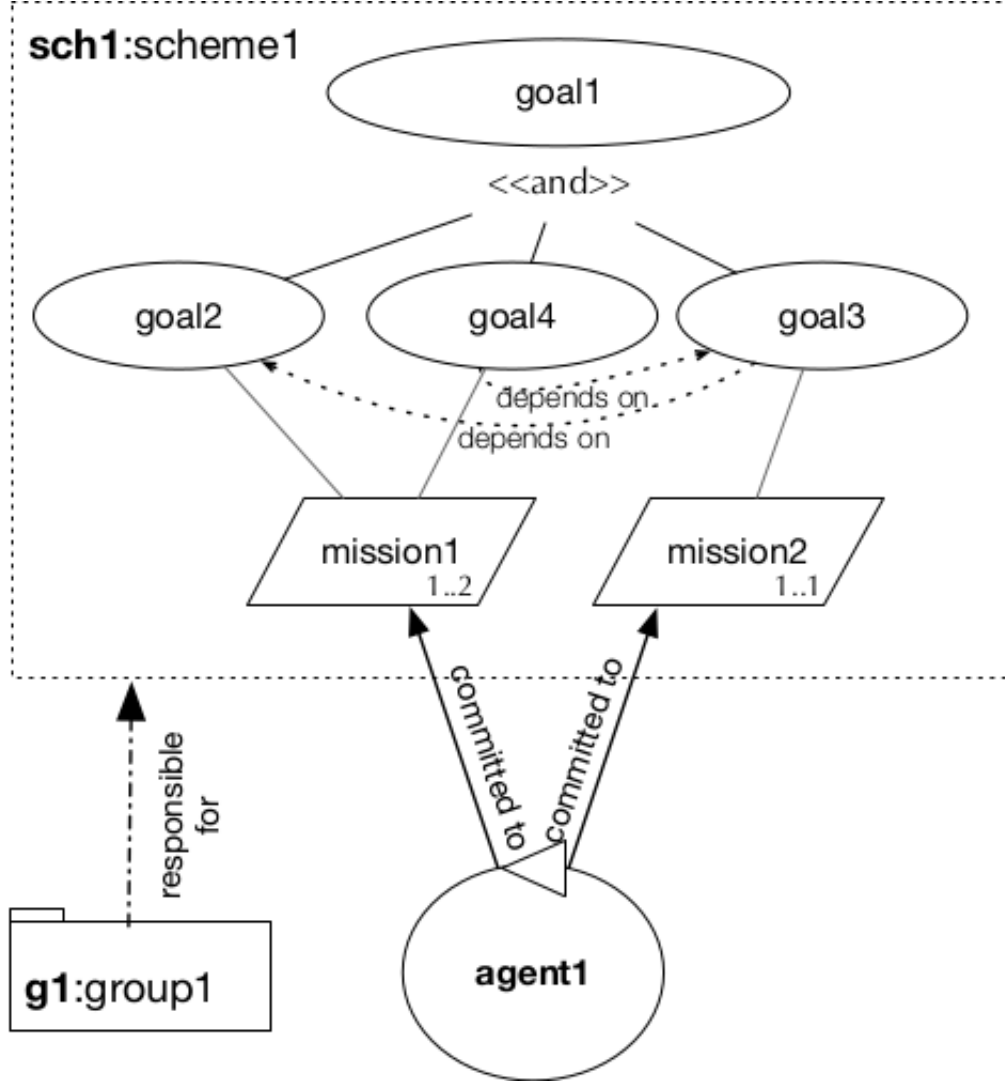
```

<structural-specification>
  <group-specification id="room">
    <roles>
      <role id="assistant" min="1" />
      <role id="controller" min="1" max="1" />
    </roles>
  </group-specification>
</structural-specification>
  
```

Functional Specification

- Specifies the expected behavior 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

Functional Specification

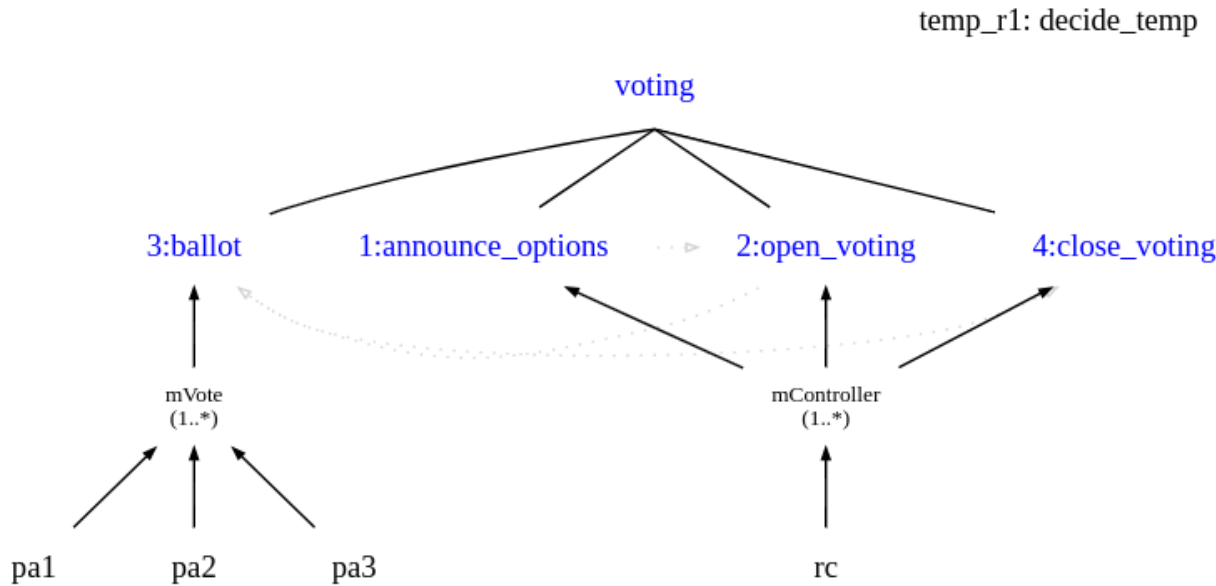


```

1  <functional-specification>
2    <scheme id="scheme1">
3      <goal id="goal1" ds="description of goal1">
4        <plan operator="sequence">
5          <goal id="goal2" ttf="20 minutes" ds="description of goal2"/>
6          <goal id="goal3" ds="description of goal3"/>
7          <goal id="goal4" ds="description of goal4"/>
8        </plan>
9      </goal>
10
11     <mission id="mission1" min="1" max="2">
12       <goal id="goal2" />
13       <goal id="goal4" />
14     </mission>
15     <mission id="mission2" min="1" max="1">
16       <goal id="goal3"/>
17     </mission>
18   </scheme>
19 </functional-specification>

```

Functional Specification Example



```

<functional-specification>
  <scheme id="decide_temp">
    <goal id="voting">
      <plan operator="sequence">
        <goal id="announce_options" />
        <goal id="open_voting" />
        <goal id="ballot" ttf="10 seconds">
          <argument id="voting_machine_id" />
        </goal>
        <goal id="close_voting" />
      </plan>
    </goal>
    <mission id="mVote" min="1">
      <goal id="ballot" />
    </mission>
    <mission id="mController" min="1">
      <goal id="announce_options" />
      <goal id="open_voting" />
      <goal id="close_voting" />
    </mission>
  </scheme>
</functional-specification>
  
```

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>  
  <!-- the norms of the application -->  
  <norm id="norm1" type="obligation"  
    role="role1" mission="mission1"/>  
  <norm id="norm2" type="permission"  
    role="role2" mission="mission2"/>  
</normative-specification>
```

Normative Specification Example

Normative Specification

id	condition	role	relation	mission	time constraint	properties
n1		assistant	<i>obligation</i>	mVote		
n2		controller	<i>obligation</i>	mController		

```
<normative-specification>
```

```
  <norm id="n1" type="obligation"  
    role="assistant" mission="mVote" />
```

```
  <norm id="n2" type="obligation"  
    role="controller" mission="mController" />
```

```
</normative-specification>
```


Organization Dynamics

In the context of Organization life-cycle

- Creation/Deletion of an Organization from an Organization specification
- Entrance/Exit of an agent
- Change of Organization specification

In the context of Organization structure life-cycle

- Creation/Deletion of a group
- Adoption/Leave of a role

In the context of Coordination activity life-cycle

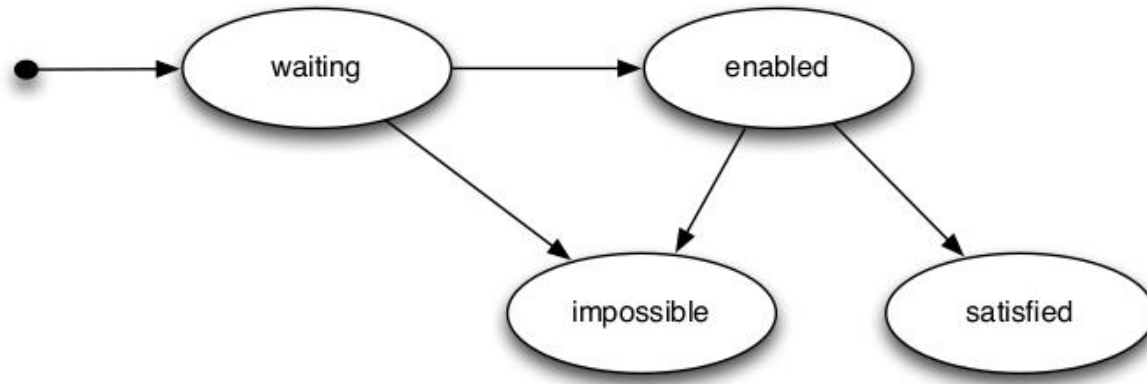
- Creation/End of a schema
- Commitment/Release of a mission
- Change of goal state

In the context of Normative Regulation activity life-cycle

- Activation/De-activation of norms
- Fulfillment/Violation of norms
- Enforcement of norms

Organization Dynamics

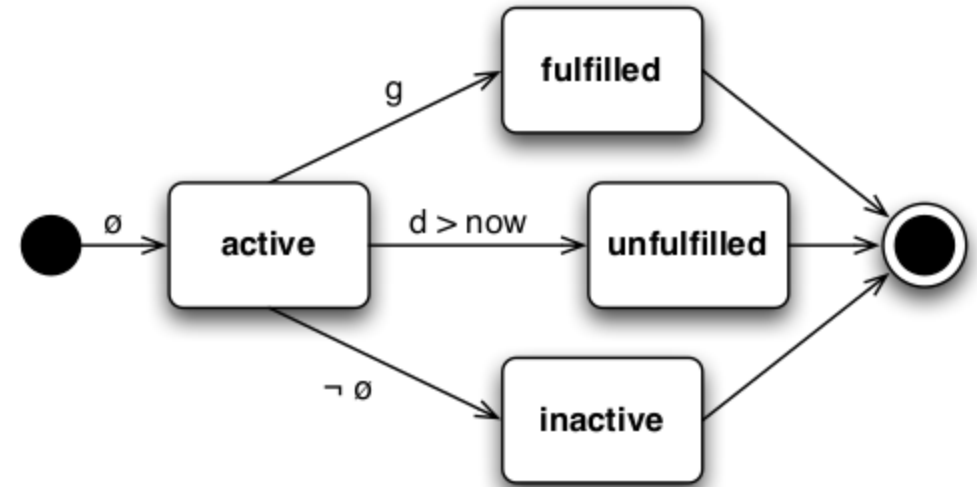
Organization Goal Dynamics



- waiting** initial state
- enabled** goal pre-conditions are satisfied and 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

Norm Dynamics



norm n : $\phi \rightarrow \text{obligation}(a, r, g, d)$

- ϕ : activation condition of the norm (e.g., play a role)
- g : the goal of the obligation (e.g., commit to a mission)
- d : the deadline of the obligation

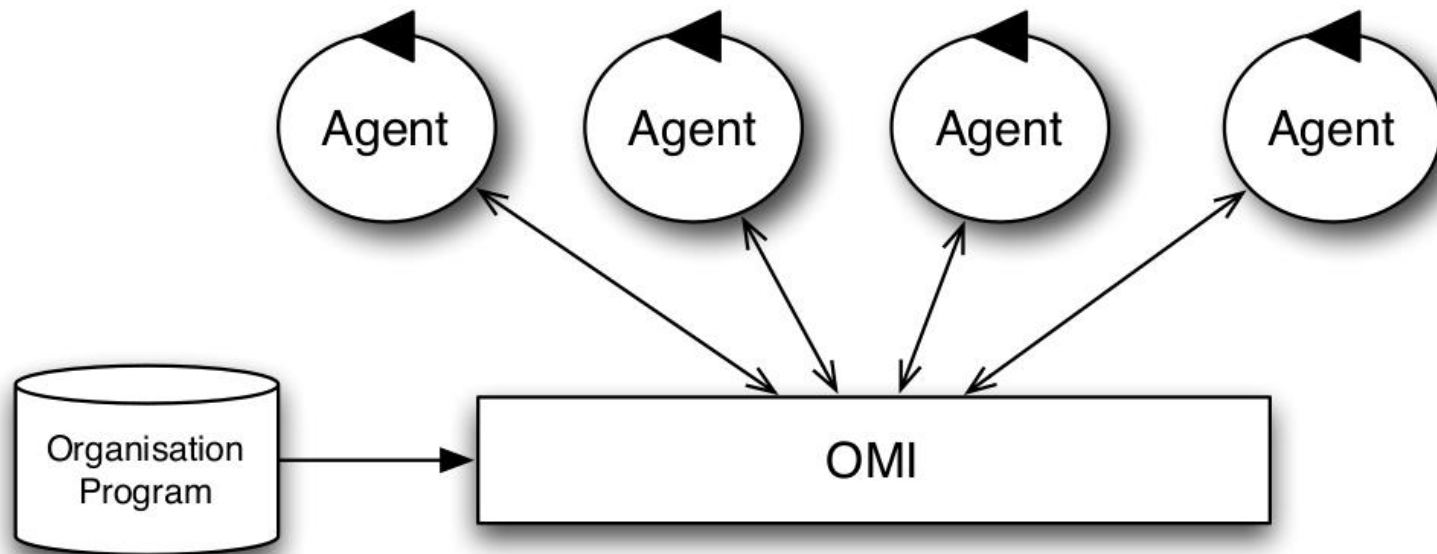
Organization Entity

`smart-room.jcm`

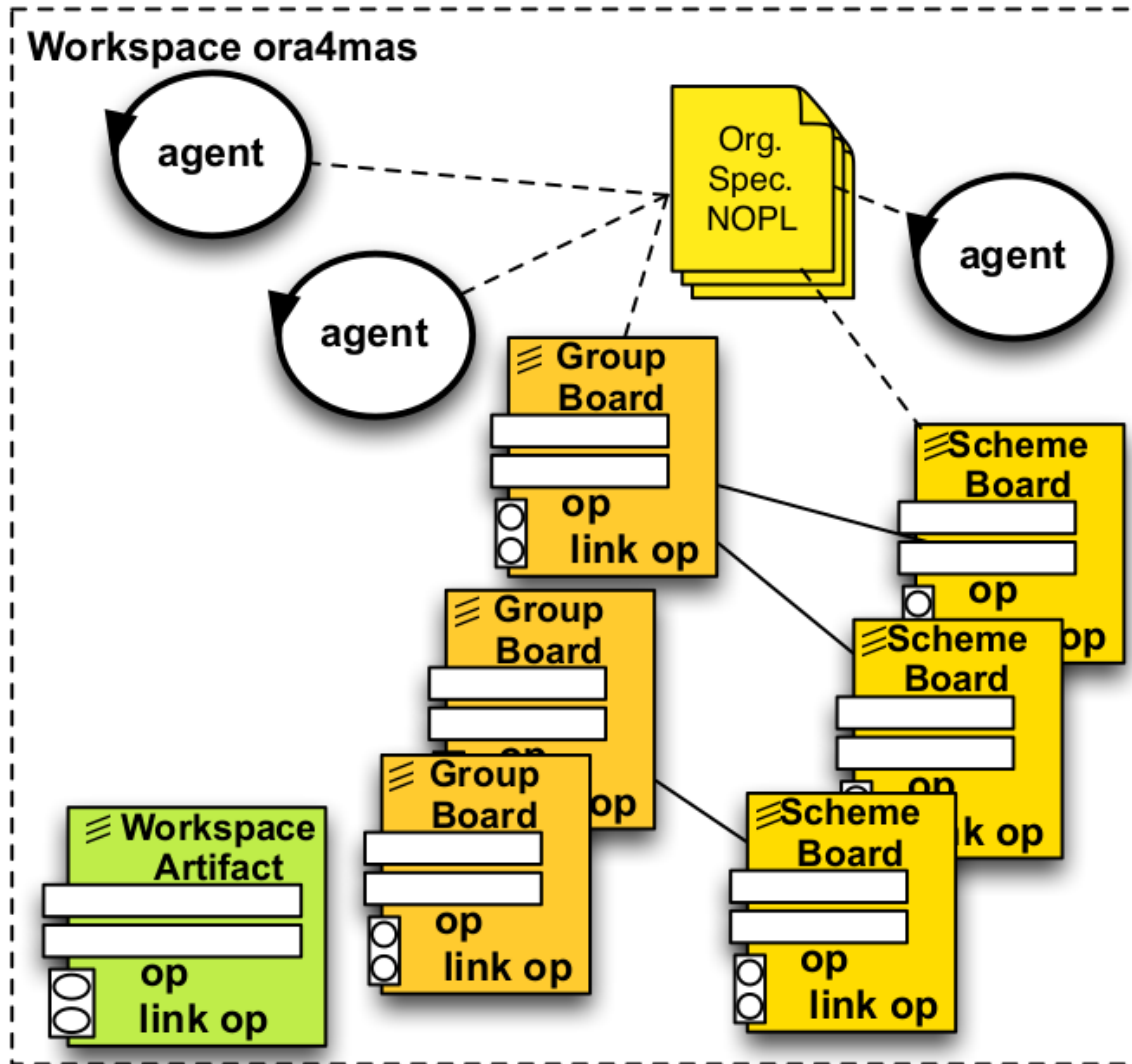
```
mas smart_room {  
    ...  
  
    organisation smart_house_org : smart_house.xml {  
        group r1 : room {  
            players: pa1 assistant  
                    pa2 assistant  
                    pa3 assistant  
                    rc controller  
            responsible-for: temp_r1  
        }  
  
        scheme temp_r1: decide_temp  
    }  
}
```

Organization Management Infrastructure (OMI)

Managing – coordination, regulation – the agents' execution within organization defined in an organization specification

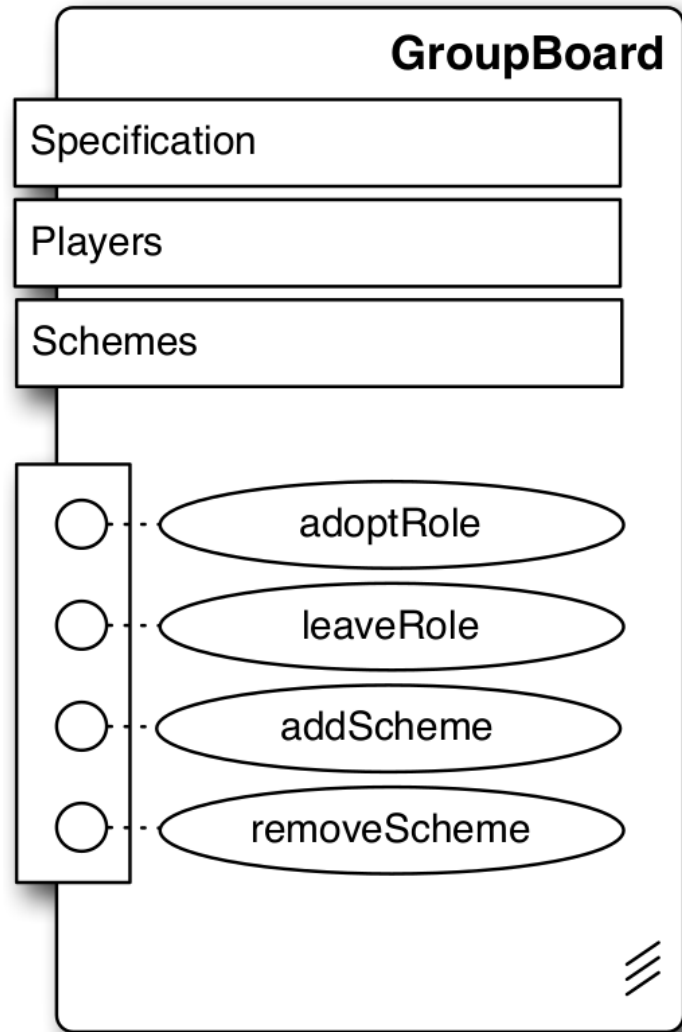


Organizational Artifacts in JaCaMo



- based on A&A and Moise
- agents create and handle organizational artifacts
- artifacts in charge of regimentations, detection and evaluation of norms compliance
- agents are in charge of decisions about sanctions
- distributed solution

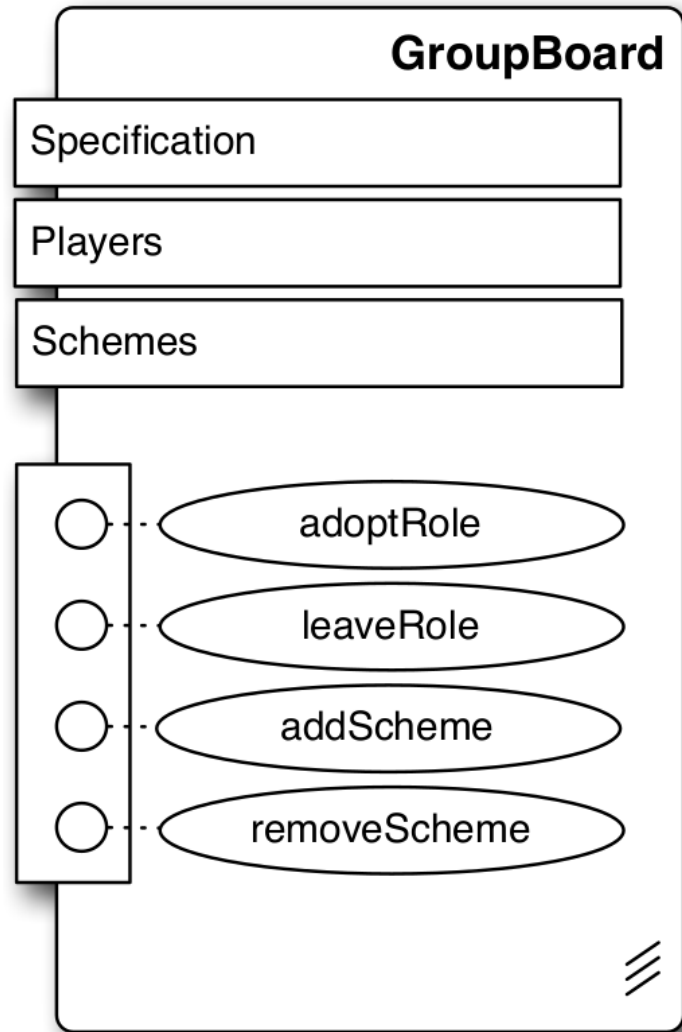
GroupBoard Artifact



Observable Properties

- **specification:** the specification of the group in the OS
- **players:** a list of agents playing roles in the group. Each element of the list is a pair (agent x role)
- **schemes:** a list of scheme identifiers that the group is responsible for

GroupBoard Artifact



Operations

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

SchemeBoard Artifact



Observable Properties

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

SchemeBoard Artifact



Operations

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

Organization Beliefs

Belief	Description
<code>play(A, R, G)</code>	Agent A is playing role R in group G
<code>commitment(A, M, S)</code>	Agent A is committed to mission M in scheme S
<code>formationStatus(S) [artifact_name(_, A)]</code>	The formation status for scheme or group A is S (possible values for S are <code>ok</code> or <code>nok</code>)
<code>goalState(S, G, LC, LA, T)</code>	Goal G , of scheme S , is in state T (possible values for T are <code>waiting</code> , <code>enabled</code> , <code>satisfied</code>); LC is a list of agents committed to the goal, and LA is the list of agents that have already achieved the goal
<code>goalArgument(S, G, A, V)</code>	Argument A of goal G has value V in scheme S
<code>obligation(A, R, G, D)</code>	Agent A is obliged to achieve G before D while R holds
<code>permission(A, R, G, D)</code>	Agent A is permitted to achieve G before D while R holds

Organization Events

Event	Description
<code>oblCreated(O)</code>	Obligation <code>O</code> was created
<code>oblFulfilled(O)</code>	Obligation <code>O</code> was fulfilled
<code>oblUnfulfilled(O)</code>	Obligation <code>O</code> was unfulfilled
<code>oblInactive(O)</code>	Obligation <code>O</code> is inactive

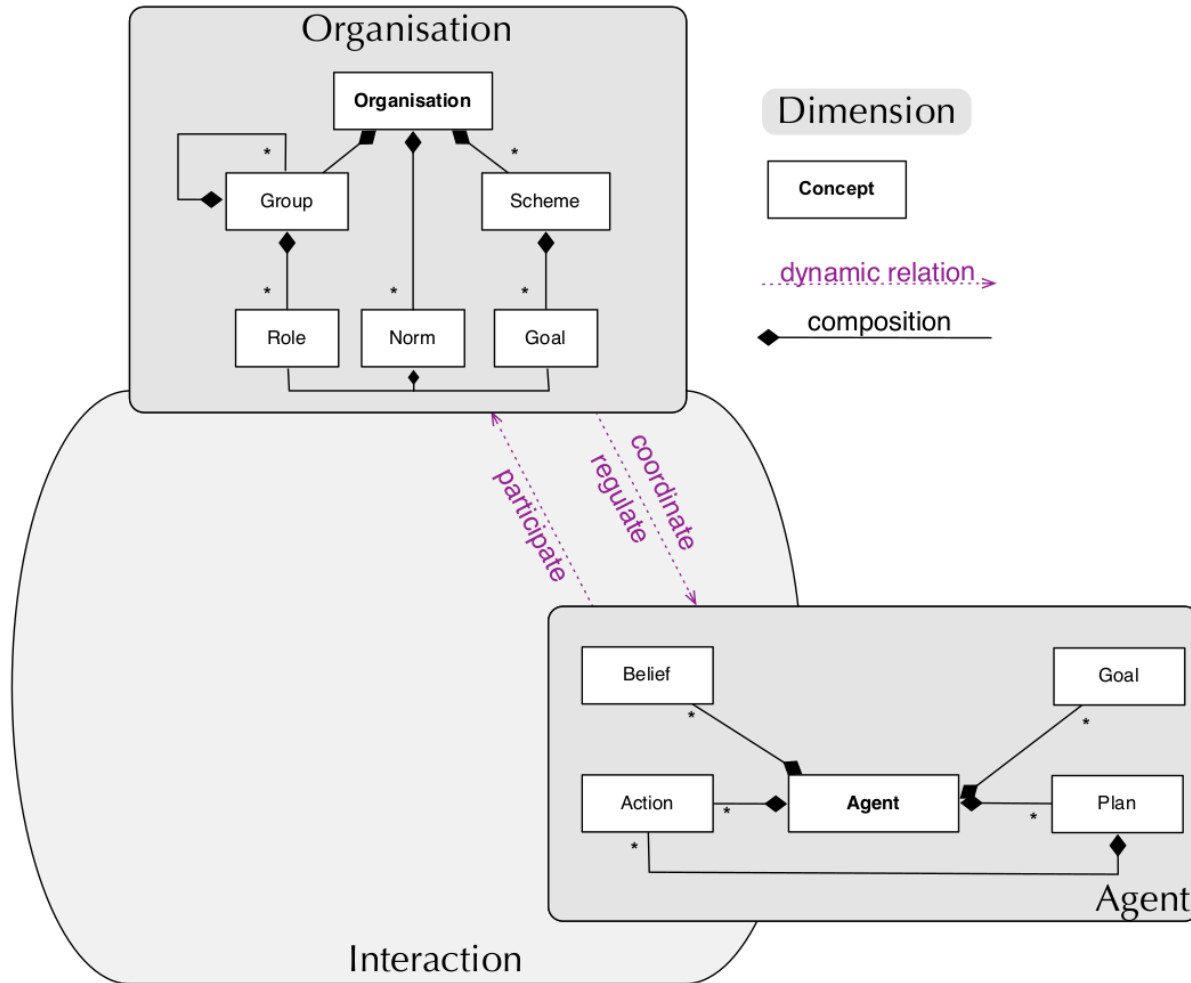
Organization Actions

Belief	Description
<code>createGroup (Name, Type, ArtId) [artifact_name (O)]</code>	Creates a new group of name <code>Name</code> , following the group specification <code>Type</code> defined in the organization specification used to create organization <code>O</code> . The organization artifact <code>GroupBoard</code> that manages it is identified by <code>ArtId</code>
<code>createScheme (Name, Type, ArtId) [artifact_name (O)]</code>	Creates a new scheme of name <code>Name</code> , following the scheme specification <code>Type</code> defined in the organization specification used to create organization <code>O</code> . The organization artifact <code>SchemeBoard</code> that manages it is identified by <code>ArtId</code>
<code>adoptRole (R) [artifact_name (G)]</code>	Adopt role <code>R</code> in group <code>G</code>
<code>leaveRole (R) [artifact_name (G)]</code>	Leave role <code>R</code> in group <code>G</code>
<code>addScheme (S) [artifact_name (G)]</code>	Add scheme <code>S</code> to the responsibility of group <code>G</code>
<code>removeScheme (S) [artifact_name (G)]</code>	Remove scheme <code>S</code> of the responsibility of group <code>G</code>

Organization Actions

Belief	Description
<code>commitMission (M) [artifact_name (S)]</code>	Commit to mission M in scheme S
<code>leaveMission (M) [artifact_name (S)]</code>	Leave mission M in scheme S
<code>resetGoal (G) [artifact_name (S)]</code>	Set goal G to not satisfied in scheme S
<code>setArgumentValue (G,A,V) [artifact_name (S)]</code>	Set V as the value of argument A of goal G in scheme S

Integrating Agent and Organization Dimensions



- Agents can interact with organizational artifacts as with ordinary artifacts by perception and action
- Agent integration provides “internal” tools for the agents to simplify their interaction with the organization:
 - maintenance of a local copy of the organizational state
 - production of organizational events
 - provision of organizational actions

Integrating Agent and Organization Dimensions

GroupBoard

```
...
joinWorkspace("ora4mas",O4MWsp);
makeArtifact(
    "auction",
    "ora4mas.nopl.GroupBoard",
    ["auction-os.xml", auctionGroup],
    GrArtId);
adoptRole(auctioneer);
focus(GrArtId);
...
```

SchemeBoard

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

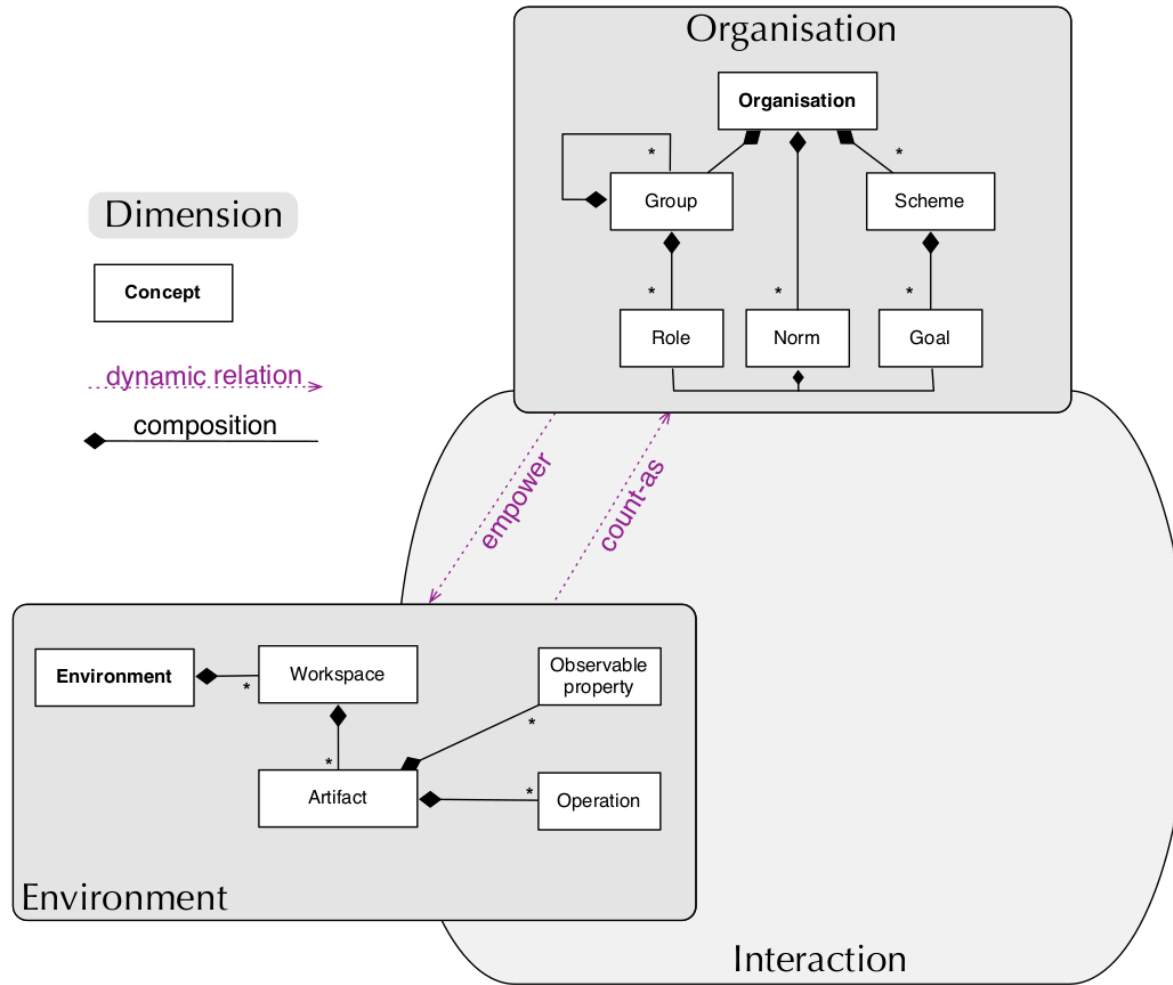
Including organization-reasoning abilities into agents

```
+play(Ag,assistant,GrId) <- .send(Ag,tell,hello).
+goalState(_,close_voting,_,_,satisfied) <- ...
```

Including norm-reasoning abilities into agents

```
+obligation(Ag,Norm,achieved(_,Goal,_),Deadline)
    : .my_name(Ag) & good(mood)
<- !Goal.
```

Integrating Environment and Organization Dimensions



- Changes in the state of the environment may **count-as** changes in the state of the organization
- This dynamic relation is a **practical way of situating organizations in an environment**, as happens for the agents, regulating some part of the environment (e.g., a traffic light at a crossroads) in a particular way and ruling it differently in other parts
- Organizations may **empower** the elements of the environment by allowing them to control and regulate actions or perception of the agents

Integrating Environment and Organization Dimensions

Constitutive norms

`X count as Y in C`

Constitutive rules

`X count-as Y`
`when Event`
`while Condition`

Example

```
play(A, assistant, "room")  
    count-as committed(A, mVote, "decide_temp")  
    while responsible("r1", "temp_r1")
```

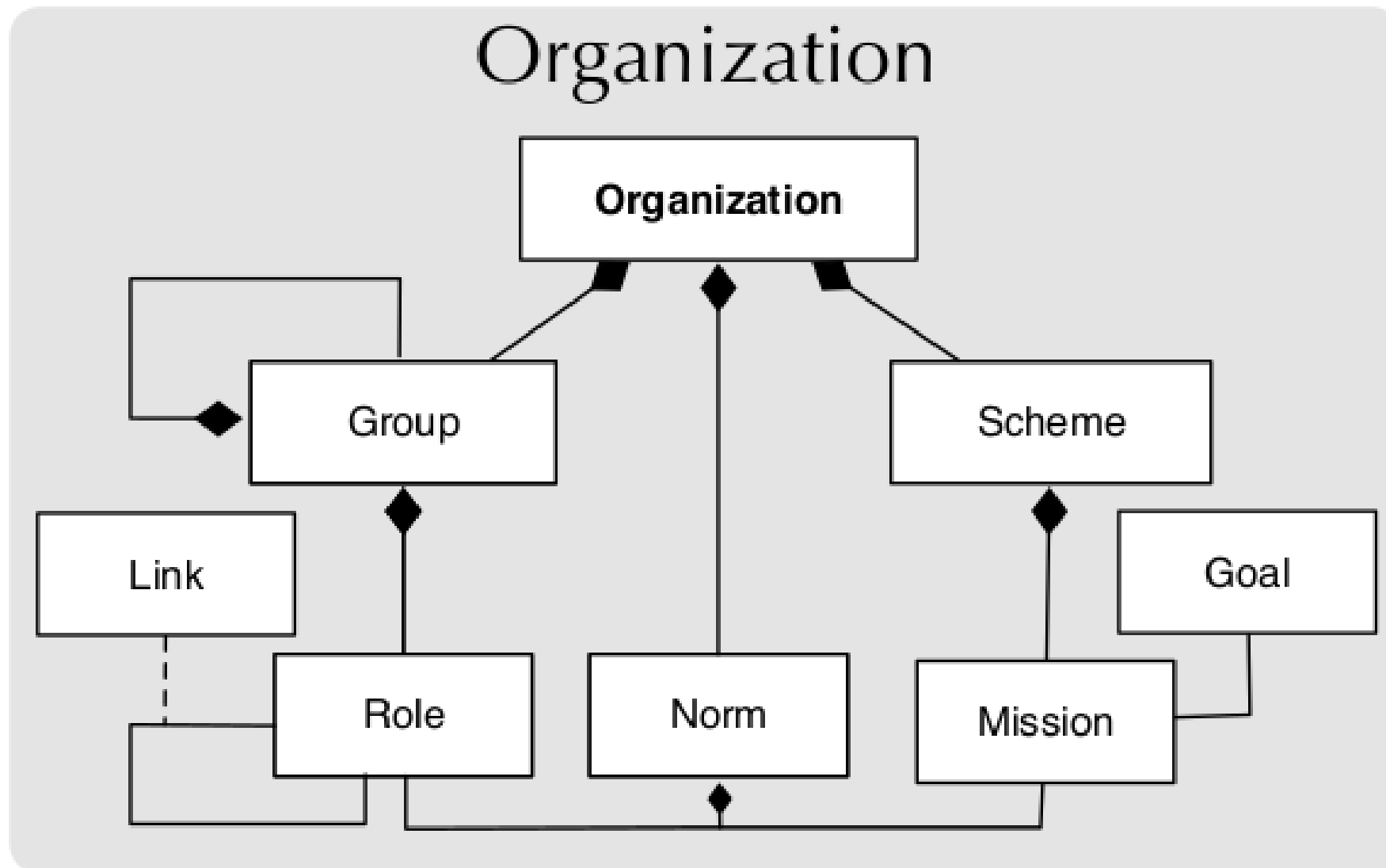
Integrating Environment and Organization Dimensions

```
institution_id : shInst.  
status_functions:  
states: play(A,R,G), responsible(G,S),  
committed(A,Mission,S), achieved(S,G,A),  
done(S,G,A).  
  
constitutive_rules:  
...  
2:  
play(A,assistant,'room')  
count-as committed(A,mAssistant,'decide_temp')  
3:  
play(A,assistant,'room')  
count-as committed(A,mVote,'decide_temp')  
4:  
play(A,controller,'room')  
count-as committed(A,mController,'decide_temp')  
...
```

Transforming organizations into situated organizations so that

- organization may act on the environment (e.g., enact rules, regimentation)
- environment may act on the organization (e.g., count-as rules) based on Situated Artificial Institution (de Brito et al., 2015)

Wrap-up: Organization Dimension



Wrap-up: Organization Dimension

- Model to specify global orchestration
team strategy is defined at a high level
- Ensure agents follow some of the constraints specified by the organization
- Help agents to work together
- The organization is interpreted at runtime, it is not hardwired in the agents' code
- The agents can 'handle' the organization (i.e., their artifacts)
- It is suitable for open systems as no specific agent architecture is required
- Organization can easily be changed by the developers or by the agents themselves

References

- Bernoux, P. (1985). *La sociologie des organisations*. Seuil, 3ème edition.
- de Brito, M., Hübner, J. F., & Boissier, O. (2015). Bringing constitutive dynamics to situated artificial institutions. In *Proc. of 17th Portuguese Conference on Artificial Intelligence (EPIA 2015)*, LNCS, vol. 9273, pp. 624–637. Springer.
- Gasser, L. (2001). Organizations in multi-agent systems. In *Pre-Proceeding of the 10th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'2001)*, Annecy.
- Hübner, J. F., Sichman, J. S., and Boissier, O. (2007). Developing Organised Multi-Agent Systems Using the MOISE+ Model: Programming Issues at the System and Agent Levels. *Agent-Oriented Software Engineering*, 1(3/4):370–395.
- Hübner, J. F., Boissier, O., Kitio, R., & Ricci, A. (2010). Instrumenting multi-agent organisations with organisational artifacts and agents: “Giving the organisational power back to the agents”. *Journal of Autonomous Agents and Multi-Agent Systems*, 20(3):369–400.
- Malone, T. W. (1999). Tools for inventing organizations: Toward a handbook of organizational process. *Management Science*, 45(3):425–443.
- Morin, E. (1977). *La méthode (1) : la nature de la nature*. Points Seuil.