* Organizations
  + Coordinate open MAS
  + Provide control of access rights
  + Allow for heterogenous agents
  + Good abstractions for modelling real world institutions

Several frameworks are available, but…

* Only a few support organizations and roles as primitives
* There is no common conceptual model of an organization
* How do agents interact with organizations and roles? (object, coordination artifact, agent)

Improve the JADE (Java Agent Development) framework

* Model of organizations and roles
  + Organizations and roles are not autonomous (act via role players)
  + Roles depend on its organization
  + Organizations and roles describe complex behaviours

🡪 Some agent-like properties

* Definition
  + Roles are entities (contain state and behaviour)
    - Role instance: role as executed by an agent (the role player)
    - Role type: specification of the role
  + Roles have goals and beliefs (role player should act like this)
  + Roles extend an agent’s abilities (“Powers”)
  + Roles can have requirements (agent needs to have certain abilities beforehand)
  + Communication influences role state (of both the speaker and the listeners)
* Roles allow encapsulation of interaction between agents, organizations, and role players
* Roles allow for
  + Structuring the organization
  + Distributing responsibilities
  + Coordination
* An agent can have multiple roles, but only a single role can be active at one time

Implementation

* Extend *Agent* class
  + - *Organization*
    - *Role*
    - *Player*
* *Role*
  + - *Role* class extensions represent role types (note how this allows organizations to play a role in other organizations)
    - Has protocols to communicate with role player
      * Request to invoke a power
      * Request to deact role
      * Tell player to execute requirement and receive the result (note that the player can refuse to execute a requirement)
      * Inform player of power execution failure / failure to receive requirement results
    - Roles can only be instantiated if there is an organization that offers it
    - Roles can modify state of its Organization
      * Roles must be on the same platform as its Organization
    - Roles can be registered in the Yellow Pages for agent search
* *Organization*
  + - *Organization* instance can offer roles
    - On Organization destruction, all its roles are destroyed
    - Has protocols to communicate with an agent who wants to play a role
      * Player informs organization that it wishes to play a role
      * Organization checks if player is authorized
      * Organization returns list of powers and requirements
      * Player checks if he can play the role (enacting) (Note: player can lie here)
* *Player*
  + - Models role playing states (enact, active, deactivated, deacted)
    - State transitions
    - Has protocols to communicate with organizations and the role

(kopieer Fig. 2a)

Finite state machine for each role played by the agent

Enact:

* Role allowed: to deactivated state
* Otherwise: to deacted state

Deactivated:

* Blocks until role is activated

Activated:

* Communication between player and role
* Player can execute requirements

Deact:

* Role is informed that player leaves role
* Data is cleaned up

**Interaction**

Requirements = goals and beliefs

**Between player and role**

* On activation of a role, player should adopt its goals and beliefs
* Programmer could invoke power directly from the role, but with this system, the agent is remains autonomous