Multi-Agent Systems

1. Characterization and Elements
2. Interactions and Interaction Situations
3. Communication
4. Interaction Specification
5. Organization
1. Characterization and Elements

A **Multi-Agent System** is a system that consists of several interacting agents.

- Distributed Problem Solving Perspective: Multi-Agent System as loosely coupled network of problem solvers for solving problems that exceed individual abilities and knowledge.

- **Conceptualization** as Multi-Agent System makes sense, if naturally pre-determined (for example due spatial or functional distribution) or simply useful for problem solving.

- **Characteristics**
  - Agents with local view and manipulation abilities
  - No global system control
  - Decentralized data
  - Asynchronous computation
Elements of a Multi-Agent System

1. A **Environment** $E$, i.e. some area usually with of some extend

2. A set $O$ of **Objects**. Objects are situated, i.e. at every point of time, a position in $E$ can be assigned to every object. Objects are (except they are agents) passive entities, i.e. they can be perceived, produced, destroyed and modified by the agents.

3. A set $A$ of **Agents**, special objects ($A \subseteq O$). They represent the active entities of the system.

4. A set $R$ of **Relations**, that relate objects (and thus also Agents) to each other.

5. A set $Op$ of **Operations**, with that agents perceive, produce, destroy or manipulate objects.

6. A set $GU$ of Operators, that represent the application of operations of $Op$ and the reaction of the world on that applications: „**Laws of the Universe**“.
Categories for characterizing MAS

- **Tasks/Problems/Goals**
  Distributed Problem Solving or pure Multi-Agent Systems
- **Number of Agents** – fixed or variable
- **Granularity** – abilities and behavior of the single agents
- **Heterogeneity** beyond architecture, implementation or hardware
- **Organisatoric distribution**
  Degree of autonomy of the agents
  Societies of individuals... hierarchies
- **Willingness to cooperation or** Antagonism
- **Complexity of their interactions**: Level of abstraction of the communication contents, range of interactions
- **Coupling**
  Strength of interactions, e.g. compare relation between communication and inference
Multi-Agent Systems

1. Characterization and Elements
2. Interactions and Interaction Situations
3. Communication
4. Interaction Specification
5. Organization
Interactions

**Interaction** happens, when several agents are put in a dynamic relation towards each other by a set of reciprocal actions. Interaction originates in a sequence of actions which’s effects influence the future behavior of the agents. Exists between agents (agent-agent-Interaction) and between agents and their environment (agent-environment-interaction).

- **Direct** by means of communication (exchange of messages)
- **Indirect** via the environment (Passives observation of the active manipulations of the environment by others)
- **Result** of Interaction can cause a change of the internal status of an agent or influence the future course of its activities
- Interactions can be distinguished by their Persistence, Pattern, Goal... (also: reactive vs. deliberative Interactions)

J. Ferber. Multi-Agent Systems, Addison Wesley, 1999
Prerequisites for Interaction

- Agents must be existing that are able to **act or/and communicate**.
- Situations must occur that give **opportunities to meet** for the agents, for example explicit cooperation, movement of vehicles that may lead to collision, shared usage of restricted resources or regulation of group formation...
- There must be **dynamic elements** for producing local and timely restricted relations between the agents: communication, manipulation, attraction, repelling, ...
- There must be some degree of freedom in the relations between the agents, that allows to maintain or cancel relations – agents need a minimum amount of autonomy
Characterization of Interaction Situations

• **Compatible and incompatible goals**
  – *Incompatible goals*: The goal of agent A is incompatible with the goals of another agent, if:
    If goals of A and B are situations p and q and p $\Rightarrow \neg q$, then: goal(A, p) $\Rightarrow \neg$ satisfied(goal(B, q)).
  – Agents are in a *cooperation situation*, if their goals are compatible

• **Relation to resources**
  Resources are all material elements of the environment that can be used for execution of an action. Every agent needs resources for the execution of its actions that happen in space and time, consume energy (cost) or need instruments as catalysts for their activities $\rightarrow$ conflicts
Characterization of Interaction Situations

• **Abilities of Agents** in relation to their tasks/goals:
  There are three types of tasks:
  – Simple tasks that an agent can execute alone
  – Complex tasks that are better executed in a team
  – Tasks that need cooperation for being executable → new functionality based on teamwork
### Categories of Interaction Situations

<table>
<thead>
<tr>
<th>Goals</th>
<th>Ressources</th>
<th>Abilities</th>
<th>Interaktion Situation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatible</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Independence</td>
</tr>
<tr>
<td>Compatible</td>
<td>Sufficient</td>
<td>Not sufficient</td>
<td>Simple Collaboration</td>
</tr>
<tr>
<td>Compatible</td>
<td>Not sufficient</td>
<td>Sufficient</td>
<td>Obstruction</td>
</tr>
<tr>
<td>Compatible</td>
<td>Not sufficient</td>
<td>Not sufficient</td>
<td>Coordinated Collaboration</td>
</tr>
<tr>
<td>In-Compatibility</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Individual competition</td>
</tr>
<tr>
<td>In-Compatibility</td>
<td>Sufficient</td>
<td>Not sufficient</td>
<td>Collective competition</td>
</tr>
<tr>
<td>In-Compatibility</td>
<td>Not sufficient</td>
<td>Sufficient</td>
<td>Individual resource conflicts</td>
</tr>
<tr>
<td>In-Compatibility</td>
<td>Not sufficient</td>
<td>Not sufficient</td>
<td>Collective conflicts over ressources</td>
</tr>
</tbody>
</table>

- Ignorance
- Cooperation
- Antagonism
Multi-Agent Systems

1. Characterization and Elements
2. Interactions and Interaction Situations
3. Communication
   - Basic aspects of communication in Multi-Agent Systems
   - Speech act-based communication
   - Specification of conversations
   - Blackboard systems
4. Interaction Specification
5. Organization
Agent Communication

• Communication partially belongs to perception (Receiving messages) and partially to action (Sending messages)

• Agent communicate for better reaching their goals (or the goals of the overall system).

• Communication enables coordination of actions for reaching a higher coherence.

• reactive agents and cognitive/deliberative agents use different forms of communication:
  – Reactive agents often just modify their environment
  – Deliberative agents intentionally generate messages
Requirements for communication

• Shared **Ontology**
  – Shared set of terms
  – Shared semantics for these terms
• Shared communication languages in **Syntax** and **Semantics**
• Technical requirements like transmission channels and transfer protocols
Syntax, Semantics and Pragmatics

- **Meaning = Combination of semantics and pragmatics**
- Influence on meaning:
  - Description – Prescription
  - Personal – conventional meaning
  - Subjective – objective meaning (external/internal effect)
  - Perspective of speaker – of listener – of society
  - Semantics – Pragmatics
  - Context of the message (Protocol)
  - Richness of the message language
  - Identity/Role of participating agents (addressee)
  - Cardinality of the receiver set
Message-based Communication

- Agents with different abilities need to communicate. ➔ different levels, the least equipped agents communicate on the lowest level.
- Role in dialogue: active, passive or both – thus can work as master, slave or peer.
- Two basic types of messages: assertion (information) and query

<table>
<thead>
<tr>
<th></th>
<th>Base-Agent</th>
<th>Passive agent</th>
<th>Active agent</th>
<th>Peer agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiving information</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Receiving queries</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Sending information</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Sending queries</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>
Communication Layers

• Specification of communication protocols usually on different layers (ISO/OSI, Internet Protocols...):
• Simple example
  – **Lowest layers** are responsible for method of connection and actual transport
  – **Middle layers** are responsible for format, syntax of the transferred information
  – **Highest levels** specify the meaning of the communicated messages.
• Binary or n-ary communication protocols
Protocols

- Protocol is **specified** by
  - Sender
  - Receiver
  - Language, that is used in the protocol
  - Encoding and decoding functions
  - Actions that shall be triggered at the receivers side
- Necessary areas of knowledge
  - Knowledge about the communication **topic, content and intention**
  - Knowledge about the communication **process**: format and semantic of the messages, dialogue conventions,...
  - Knowledge about the concrete communication **sequence**, i.e. the current dialogue
- Speech Acts as building blocks
Speech Acts

• Communication as a form of action
• The articulation of a statement changes the environment: “I condemn you to 6 month arrest“ – Speaker changes the environment, here: social status of the arrested person.
• Speech Acts are often characterized by „Hereby...“, ...
Three Aspects of a Speech Act

- **Locutionary** Component: the actual generation of the utterance, the writing down the sentence – the production of the sentence using a given grammar/lexicon

- **Illocutionary** Component: the intention of the speaker when executing the speech act.
  - Studies in language pragmatics: „Illocutionary force“ (affirm, promise,...) and content of the statement, which is the object of the illocutionary force.
  - Representation in form of F(P). Examples: query(it rains), assert(it rains) → **performative**(content)
  - Intention of a (human) speaker is not easy to understand: „I‘m cold“

- **Perlocutionary** Component: Effects of the speech act onto the state of the receiver (modification of beliefs, actions..)
Main Types of Speech Acts

Representation of the illucotionary layer of an utterance in form of performatives

• **Assertive Acts („Representatives“)**: Express information about the world by explanation (e.g. „rectangles have four right angles“)

• **Directive Acts**: command to the receiver (e.g. „Wash your hands“, „What is the third decimal of $\pi$?“)

• **Promissive Acts („Commissives“)** commitment of the sender, to execute certain actions in the future (e.g. „I promise to write a postcard“)

• **Expressive Acts** serve for information the receiver about the mental state of the sender (e.g. „I’m sorry for yesterday“)

• **Declarative Acts**: The utterance is equivalent with an action („I curse you“, „I declare the meeting as opened“...)
Success Requirements for a Speech Act

- Abilities for sending and receiving speech acts
- Syntax of the speech act is correct
- Prerequisites – e.g. B is able to execute P, A believers that B is able to execute P, neither A nor B are sure about that B will execute action P.
- Conditions of seriousness
- Conditions of the content (goals of agents)
- “degree of power“ – Intensity of the execution of the speech act.
- Knowledge Query and Manipulation Language
- Protocol should be shared by all agents, shall be precise and just use a restricted number of communication primitives.
- Separation between semantics of the protocol (usually domain-independent) and semantics of the message content (usually domain specific)
- Idea: All information that are necessary for understanding the content are contained in the communication.

- KQML-Performatives are based on speech act performatives – Semantics of the performatives is not depending on the domain.
Basic structure

(KQML-Performativ
  :sender  <word>
  :receiver <word>
  :language <word>
  :ontology <word>
  :content <expression> ...)

KQML distinguishes between three constituents

- **Content level**: Message content with the field:
  :content (actual message) :language (language of content) and :ontology (vocabulary of the content)

- **Message level**: specifies type of speech act, position in protocol by :in-reply-to and :reply-with. (:reply-with is a field for the sender, :in-reply-to for the receiver for representing the relation between messages). There may be additional fields depending on the performative.

- **Communication level**: Parameter like sender, receiver and message-id

KQML provides a „wrapper“ for a message.
Basic categories of KQML performatives:

- **Basic requests** (evaluate, ask-one, ask-all...)
- **Request performatives with multiple answers** (stream-in, stream-all,...)
- **Answer primitives** (reply, sorry...)
- **Generic information performatives** (tell, achieve, cancel, untell, unachieve,...)
- **Generator performative** (standby, ready, next, rest,...)
- **Performative concerning abilities** (advertise, subscribe, monitor...)
- **Network primitives** (register, unregister, forward, broadcast....)

→ 41 primitives in 1993
• Simple example

(tell
  :sender Agent1
  :receiver Agent2
  :language KIF
  :ontology BlocksWorld
  :content (AND (Block A) (Block B) (On A B))
)

• Messages can also be nested within other messages:
Example: Agent1 cannot directly communicate with Agent2, therefore it asks Agent 3 to forward a message to agent Agent2:

(forward
  :from Agent1
  :to Agent2
  :sender Agent1
  :receiver Agent3
  :language KQML
  :ontology kqml-ontology
  :content (tell
    :sender Agent1
    :receiver Agent2
    :language KIF
    :ontology BlocksWorld
    :content (AND (Block A) (Block B) (On A B))
  ))
(evaluate
   :sender Agent1  :receiver Agent2
   :language KIF   :ontology motors
   :reply-with q1  
   :content (val (torque m1)))

(reply
   :sender Agent2  :receiver Agent1
   :language KIF   :ontology motors
   :in-reply-with q1 
   :content (= (torque m1) (scalar 12 kgf))

(stream-about
   :sender Agent1  :receiver Agent2
   :language KIF   :ontology motors
   :reply-with q1  
   :content m1)

(tell
   :sender Agent2  :receiver Agent1
   :in-reply-with q1 
   :content (= (torque m1) (scalar 12 kgf))

(tell
   :sender Agent2  :receiver Agent1
   :in-reply-with q1 
   :content (= (torque m1) normal)

(eos
   :sender Agent2  :receiver Agent1 
   :in-reply-with q1)
(advertise
  :sender Agent2  :language KQML  :ontology K10
  :content (subscribe
    :language KQML  :ontology K10
    :content (stream-about
      :language KIF  :ontology motors
      :content m1 )))

(subscribe
  :sender Agent1  :receiver Agent2  :reply-with s1
  :content (stream-about
    :language KIF  :ontology motors
    :content m1 ))

(tell
  :sender Agent2  :receiver Agent1
  :in-reply-with s1  :content (= (torque m1) (scalar 12 kgf))
(tell
  :sender Agent2  :receiver Agent1
  :in-reply-with s1  :content (= (torque m1) normal)
(untell
  :sender Agent2  :receiver Agent1
  :in-reply-with s1  :content (= (torque m1) (scalar 12 kgf))
(tell
  :sender Agent2  :receiver Agent1
  :in-reply-with q1  :content (= (torque m1) (scalar 15 kgf))
(eos
  :sender Agent2  :receiver Agent1  :in-reply-with q1)
KQML – Criticism

• The set of KQML-performatives is **neither closed nor minimal.** A KQML-Agent can choose which primitives it wants to deal with or not.

• Introduction of Facilitators (Mediator agents): Agents able to communication using KQML and provide a lot of useful services: Registering of services, message forwarding, Naming Service, Switchboard...

• Main problems:
  – huge, „ad hoc“ set of primitives, but none for commitments
  – Missing definition of semantics of performative → no clear definition what is actual KQML resp. the meaning of its performatives! (different attempts to add formal semantics afterwards)
  – No prescription, how KQML and transfer mechanisms shall be implemented.
Next Step: FIPA-ACL

- ACL (Agent Communication Language)
  - Syntax is identical to KQML: equal structure, equal message field (reuse of infrastructure like parsers... when replacing KQML by FIPA-ACL)
  - Formal Specification of semantics of Communication Acts (= Performative) based on „SL“ (multimodal logik with operators for Beliefs (B), Desires(D), uncertain beliefs (U) and intentions (I))
    - „feasibility condition“ = pre-condition, what has to be true for a successful speech act
    - „rational effect“ = post-condition, what the sender hopes about the effect of the speech act
  - 22 Primitives:
    „purer“ ACL – no CAs for switch board, registration...
<table>
<thead>
<tr>
<th>Performative</th>
<th>Passing information</th>
<th>Requesting information</th>
<th>Negotiation</th>
<th>Performing actions</th>
<th>Error handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept-proposal</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agree</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Cancel</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Cfp</td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Confirm</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disconfirm</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Inform</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inform-if</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inform-ref</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not-understood</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Propagate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Propose</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Proxy</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Query-if</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query-ref</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refuse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Reject-proposal</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Request-when</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Request-whenever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Subscribe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
• **accept-proposal**  Agent accepts proposal of the other
• **agree**  positive answer to a request
• **cancel**  an earlier request is not pursued any more, the agent that has send the request, also sends the cancel
• **cfp**  „call for proposals“, Start of a negotiation: „this is the task and conditions for its execution, send proposals to me “
• **confirm**  confirms the truth of a statement, if the sender beliefs that the receiver is not sure about.
• **disconfirm**  similar to confirm, confirms that statement is not true.
• **failure**  Sender informs that the execution of the requested action has failed
• **inform**  basic primitive for notifying about information. Content is a statement that the sender wants the receiver to believe
• **inform-if**  typically in content of a request message. Tell me whether content is true or false.
• **inform-ref**  similar to inform-if, asking about a value.
• **not-understood**  A agent does not understand the action of another one, typically a received message. Content: not understood message and justification why not understood. Central element of error handling
• **propagate**  Content of a message consists of another message and description of a set of agents the message should be forwarded to
• **propose**  answer to **cfp** containing proposal
• **proxy**  Sender takes receiver as a proxy for a set of agents. Content is a message and a set of agents
• **query-if**  an agent asks another one whether a statement is true or false
• **query-ref**  an agent asks another for a value for a given expression
• **refuse**  message that an agent will not execute an action
• **reject-proposal**  negative answer to a **propose**
• **request**  basic primitive for requesting the execution of an action
• **request-when**  content is an action and a statement. When statement is true, the action shall be executed
• **request-whenever**  action shall be executed whenever a statement becomes true
• **subscribe**  the sender wants to be notified when there is a change in a statement
Semantics, Inform, Request

- Inform and Request are most important primitives, all other can be expressed based on them
- **Inform**: content \( \varphi \) is a statement
  \[
  \langle i, \text{inform}(j, \varphi) \rangle
  \]
  feasibility condition: \( B_i \varphi \land \neg B_i (Bif_j \varphi \lor Uif_j \varphi) \)
  rational effect: \( B_j \varphi \)
- **Request**: content \( \alpha \) is an action
  \[
  \langle i, \text{request}(j, \alpha) \rangle
  \]
  feasibility condition: \( B_i \text{Agent}(\alpha, j) \land \neg B_i I_j \text{Done}(\alpha) \)
  rational effect: \( \text{done}(\alpha) \)
Practical Aspects of KQML/ACL

• Every system that uses KQML/ACL needs:
  1. Collection of APIs for **assembling**, **sending** and **receiving messages**
  2. **Infrastructure for services** – Whitepage server, registering, switchboard services, yellow pages... (How to find an agent with specific abilities?...)
  3. **Code for every Performative**, for executing the actions that are described in the semantics definitions

• Ideally a developer shall only need to realize 3. Others as reusable components available in class libraries, platforms, etc.

• Successful communication also needs content languages and ontologies
Multi-Agent Systems

1. Characterization and Elements
2. Interactions and Interaction Situations
3. Communication
4. Interaction Specification
   - Conversations
   - Cool
   - Petri Nets
   - AUML
5. Organization
Conversation

- When an agent sends a message, it possesses expectations how a answer could look like
- high-level structure of agent communication:

A conversation is a pattern of message exchange between two or more agents that agree on following this pattern.

- Conversation as a collectively generated sequence of messages – e.g. negotiations, auctions...
- Conversation is connected to a task – Registration, negotiation ...
- Conversations are guided by a priori specified protocols
• Necessary:
  – Specification of conversation
    • Roles of the participants,
    • Constraints and dependencies between single messages
    • Sequence of messages:
      – Finite state automaton
      – Petri-Nets
      – AUML
  – Shared usage of a conversation
• **Set of conversations, that an agent can participate in is the abstract interface of an agent**
• Explicit formal specification enables verification of the protocol directly and of the agents based on such protocols
Conversation Specification with COOL

- **COOL (COOrdination Language)**
- Extended finite state automata
- Nodes as states,
- Links as state transition based on receiving or sending a message
- For ever agent/role specific automaton
- **Problem**: Agents follow different automata for participating in the same conversation.
- Agent can participate in more than one conversation in parallel → identification of conversation the message belongs to.
- Rule-based selection with which conversation the agent is currently progressing – context of a conversation
COOL: Specification of a conversation

(def-conversation-class
  'customer-conversation
  :name 'customer-conversation
  :content-language list
  :speech-act-language 'kqml
  :initial-state 'start
  :final-states
    '(rejected failed satisfied)
  :control
    'interactive-choice-control-ka
  :rules '(
    (start cc-1)
    (proposed cc-13 cc-2)
    (working cc-5 cc-4 cc-3)
    (counterp cc-9 cc-8 cc-7 cc-6)
    (asked cc-10)
    (accepted cc-12 cc-11)) )

(def-conversation-rule 'cc-4
  :current-state 'working
  :received
    '(ask :sender logistics
      :content (finished ?item))
  :next-state 'accepted
  :transmit
    '(tell :sender customer
      :receiver logistics
      :content (accepted ?item)
      :conversation ?convn)
  :do
    '(put-conv-var ?conv ?item
      (cadr (member :content ?message)))
  :incomplete nil)
Specification using Petri Nets

- Established mean for representing concurrent processes
- Petri Net directed graph with places, transitions, flow relation: \( N = (S,T,F) \) mit
  - \( S \cap T = \emptyset \), (bipartite Graph), \( S \cup T \neq \emptyset \),
  - \( F \subseteq (S \times T) \cup (T \times S) \), \( \text{dom}(F) \cup \text{cod}(F) = S \cup T \)
- Extension using token → Condition Event System
  - Constellation of tokens represents current state of the net during simulation.
  - Places („Conditions“) store token,
  - Transitions („Events“) transfer them to other places.
  - Are tokens at all/one input places, then the transition may fire:
    - One token is deleted from every input place.
    - Token is held for a limited time interval („timed petri nets“)
    - To every (one) output place a token is added
  - Stochastic Petri Nets, Coloured Petri Nets
Petri Nets and Conversation
Interaction Specification with AUML

• AUML extends the interaction diagrams of standard UML 1.x to include roles, decision points, concurrency, modularity and multicasting

• Introducing Agent Interaction Protocols (AIP), so that these extend the sequence diagrams by
  – Role for agents are introduced.
  – multithreaded lifelines an agent can go through a protocol following different paths (And/Or Graph).

• *thread of interactions* for sending 'OR', 'AND' or 'XOR' messages.

• Nested protocols or overlapping protocols
Extended UML message semantics

• Every message (arrow) possesses a `message label` with the following elements:
  – Communication act
  – Cardinality (to how many agents)
  – Conditions
  – Number or repetitions
protocol templates

- Protocols can be parameterized and reused
Current Status

• Extending UML for Agents was a project supported by many groups → www.auml.org

• See for example the complete suggestion for extended sequence diagrams:

• FIPA used AUML for specifying the protocol standards

• But: currently very silent, most concepts are realized in UML 2 and OMG SysML
  – UML 2 includes loops, alternatives, parallelism, sequences and critical fragments
AUML – UML 2.0

Blackboard Architecture

**Backboard Metaphor**: A group of experts (problem solver) regard a blackboard on the wall on that the current status of a problem solution is depicted. Every agent is able to recognize its potential contribution, the agents executes the problem solving activities and communicates its results via the table.
Blackboard Architecture

- Independence of expertise: no mutual support, knowledge about other agents necessary
- Integration of different problem solving techniques
- Flexible representation on blackboard
  - Application specific,
  - Must be understandable by the agents
  - Evaluation of contents of the blackboard?
- Every agent reads and writes into shared memory → Coordination necessary
- Synchronous and asynchronous working possible
- Addressee does not need to be known
- Agent decides when and what it announces, when it searches for information and how he evaluated this information
Extensions

• Regionalization of the blackboard possible
• Event-based activation of experts possible
• Necessity of control (ideally using strategical knowledge)
• Enables incremental generation of solutions
  – Agents come forward based on events
  – Appropriate agents are selected
  – Their contributions cause changes on the blackboard
Multi-Agent Systems

1. Characterization and Elements
2. Interactions and Interaction Situations
3. Communication
4. Interaction Specification
5. Organization
   - Cooperation
   - Organizational View
   - Examples for Organizational Modeling: AGR, MOISE+
   - Relations
   - Organizational Structures
Beyond Communication

• Interaction in MAS is not based on Communication alone
• Organizational View
  – MAS is more than a collection of interacting agents, there is a meaningful “together”
  – MAS engineering requires high level agent-independent abstractions
  – Explicit social concepts, defining the society in which agents participate
An **organization** is an arrangement of relations between components or individuals that produce a unity or system that possesses properties that are not existent on the level of components or individuals.

→ The organization mutually connects different elements, events or individuals so that the elements become components of a whole.

- **Boissier**: Purposive supra-agent pattern of emergent or (pre-)defined agent cooperation
- **Gasser**: An **organization** provides a framework for activity and interaction through the definition of roles, behavioral expectations and authority relationships (e.g. control)
Agents and Organisation

Organization

Emergence of Properties

System of Interactions

Restrictions Social Goals

Agents
Perspectives on Organisations

Agent Centred

Agents don’t know about organisation

Organisation Centred

Organisation Specification
Organisation Entity

Local Representation
Observed Organisation

Designer / Observer
Bottom-up
Top-down

EASSS 2010 Tutorial of O. Boissier
Main Properties

1. An **Organization** consists of agents that produce the organization's behavior.

2. The overall organization can be divided into partitions, that may be overlapping: **Groups**.

3. Functional behavior of the agents is in relation to the activities of the overall organization.

4. Agents participate in dynamic relations (pattern of activity), that are characterized by roles, tasks or protocols describing a kind of super-individual.

5. Behavioral types are connected via relations between roles, tasks and protocols.

---

Why „Organization“?

- It guarantees a relatively high degree of reciprocity dependencies and reliability, thus enables the system to be persistent over a certain time, also when there are random disturbances.
- Security and **predictability**
- Organization as **Architecture** – Stability versus Flexibility
- Integration View – Organization produces **Constraints** for the relations and interactions of the agents
- High-level division of labor – Distribution of activities to Roles on a aggregated level
- Often recursive constitution
Levels of Organization

1. **Micro-social** Level refers to
   - Interactions between agents
   - Different types of connections between a small number of agents

2. **Group Level** refers to
   - Mediating structures, that play a role constructing a complete organization.
   - Differentiation of Roles and Activities of the agents,
   - Origin of organizational structures between agents
   - Considers problem of aggregation of agents during constitution of an organization

3. **Global Organization** refers to
   - Dynamics of a large number of agents
   - All over structure of the system and its development
Analysis / Design of Organizations

• **Top-Down-Procedure**
  – Starting Point: Organization as a whole and its desired properties
  – Attempt to derive architecture and behavior of the agents from that aggregated properties
  – Imposed

• **Bottom-Up-Procedure**
  – Starting Point: Definition of the agent
  – Attempt to find desired properties of the organization as a consequence of the interactions between the agents
  – Emergent / Evolutionary

A Priori versus A Posteriori
Organization Structure and Organization

- Persistent parts of an organization = the remainders when taking away components or individuals that enter and leave the organization → relations that make the aggregate to a whole
  = **organization structure**
- Characterizes a class of concrete organizations on an abstract level
- A (concrete) organization in an **Instance** of an organization structure.
Aspects of Organization Structures

• **Structural Aspect**
  – Partitioning Structure: How agents are combined to **groups**, how groups are interconnected
  – **Role** Structures: for every group: a set of roles, their relations and the connected constraints

• **Dynamic Aspect** → Institutionalized Interaction Pattern
  – Modalities to produce, leave, adopt role...
  – How modalities are applied and how commitments and permissions are controlled
  – How partitioning and role structures related to agent behavior.
Role

- Characterizes the functions that an agent has to fulfill, resp. the services that the agents has to offer → Abstract description of the expected behavior of an agent
- describes
  - Constraints (Commitments, prerequisites and skills), that an agent has to fulfill for taking the role
  - Gain (Abilities, Authorization, Profit), that an agent gets for playing a role.
  - Responsibilities, connected with the role
- Position of agents within an organization, together with the behavior that is expected from them for the organization fulfilling its goals
- Place holder for representation of interaction pattern that an agent is participating in when playing the role
- Independent from the internal structure of the agent.
Relations

- Type/Function of a relation
- Control Structures
- Intensity of coupling
- Formation
- Redundancy
- Degree of Specialization

J. Ferber. Multi-Agent Systems, Addison Wesley, 1999
# Type of Relations

<table>
<thead>
<tr>
<th>Relation</th>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquaintance Relation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication Relation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subordination Relation</td>
<td>Master/Slave</td>
<td></td>
</tr>
<tr>
<td>Operative Relation</td>
<td>Dependencies between tasks</td>
<td>Commitment to do something</td>
</tr>
<tr>
<td>Informational Relation</td>
<td>Dependencies for knowledge</td>
<td>Commitment to validate something</td>
</tr>
<tr>
<td>Conflictual Relation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Competition Relation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Principles

1. The **Organizations Level describes what, not how**. It handles the structures in the patterns of agent activities, but does not describe agent behavior. No code, just specification, norms or laws, constraints and expectations on the agent behavior
   → what remains when take out the actual agents

2. Organization as mean for partitioning the system – every **group represents context for interaction** between agents → group as organizational unit, within that agents can freely communicate in a shared language. Groups establish borders...

During the last years, several frameworks for the (formal) description or specification of organizations have been developed:→ in combination with AOSE methods, but also stand alone. Examples: AGR, MOISE+, OperA, AUML, ISLANDER....
AGR (Agent/Group/Role)

- Organization is solely described on **basis of its structure**: groups and roles are arranged to form a whole.
- **Agents** are black boxes, only specified as communicative entity that plays a role within a group.
- Every agent is member of **at least one group**.
- **Groups** are atomic sets of agents; each agent part of n groups, that freely overlap; **Two agents may only communicate if and only if they belong to the same group**.
- Any agent can found a group, founder plays manager role.
- **Roles as abstract representation of agent function**; roles are **local** in a group, an agent may handle multiple roles. Abstract communication schemes are defined for roles.

→ MADKIT is based on AGR

Basic Meta-Model of AGR
Describing Organizational Structures
Specific Organization

• "Cheeseboard" diagram
Organizational Dynamics

- Creation of groups
- Entering of groups
- Interaction
- Life of agent in several segments of same color
- Variant of UML sequence

Diagram:

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role 1</td>
<td>Role 2</td>
</tr>
<tr>
<td>Role 3</td>
<td></td>
</tr>
</tbody>
</table>

Time:
- Creating a group
- Leaving a role
- Entering a group
Modeling Organizations using MOISE+

- MOISE+ (Model of Organization for multi-agent SystEms)
- Structural Aspects → Groups, Roles, Links
- Functional Aspects → Global Goals, Decomposition to Plans and Distributions to Agents (specified by social schema)
- Deontic Aspects → Permissions and Obligations of a role to a mission = Connecting both modeling views

Structural Specification

- **Individual Level** \(\rightarrow\) Roles with inheritance relation
- **Social Level** \(\rightarrow\) Links: 
  \[link(p_s, p_t, type)\]
  - acq = Acquaintance
  - com = Communication
  - aut = Authority
- **Collective Level**
  - Relations of compatibility between roles
  - Group Specification:

\[gt = \text{def} (R, SG, L_{\text{intra}}, L_{\text{inter}}, C_{\text{intra}}, C_{\text{inter}}, np, ng)\]

- \(R\): set of all roles,
- \(SG\): Set of all sub-groups,
- \(L_{\text{intra}}, L_{\text{inter}}\) set of all links,
- \(C_{\text{intra}}, C_{\text{inter}}\) Role compatibilities,
- \(np\) – \((\text{min}, \text{max})\) number of agents with role,
- \(ng\) – Subgroup-Cardinatity
Functional Specification

Social Schema → Goal Decomposition Tree

- **Missions** – Set of global goals
- Global plans
- Agent commits to mission
- combination of plans with 3 operators
  - Sequence
  - Choice
  - Parallelism
- Goal Description List
- Definition of a social schema:
Example

score a goal

- get the ball
  - go towards the opponent field
    - be placed in the middle field
      - be placed in the opponent goal area
        - kick the ball to (agent committed to m2)
      - go to the opponent back line
        - kick the ball to the goal area
          - shot at the opponent’s goal
            - m1, m2, m3
  - m1

Key

Organizational Entity

- Lucio --------- m1
- Cafu --------- m2
- Rivaldo ------ m3

Scheme:
- missions
- goal
- success rate

Sequence, Choice, Parallelism