Engineering Multi-Agent Systems I

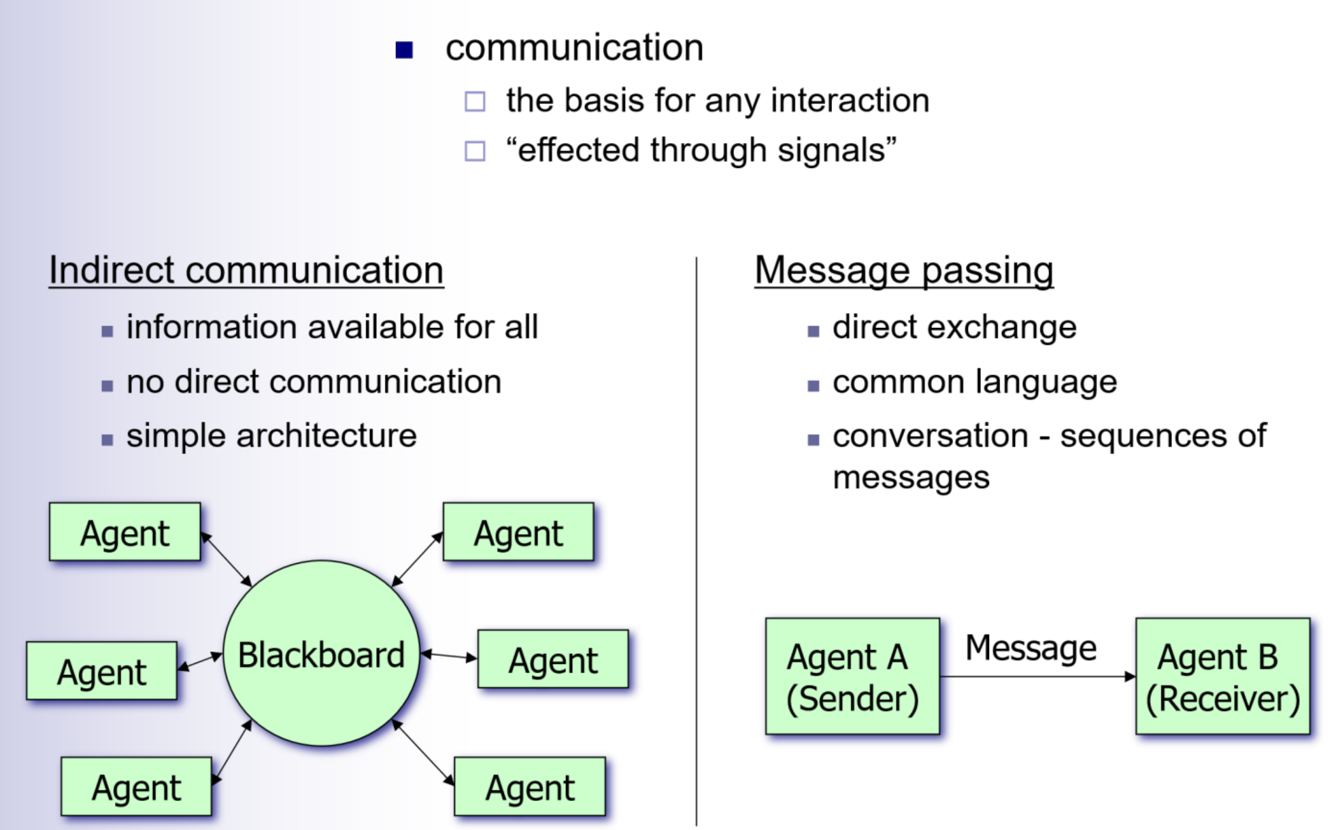
PCS-5045

Escola Politécnica da USP
LTI – Laboratório de Técnicas Inteligentes

Agenda

- Agent Communication
- MAS as Distributed Systems
- Programming MAS

Agent Communication



- communication
  - □ the basis for any interaction
  - □ "effected through signals"

Indirect communication

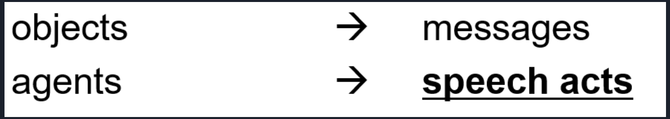
- information available for all
- no direct communication
- simple architecture

Message passing

- direct exchange
- common language
- conversation - sequences of messages

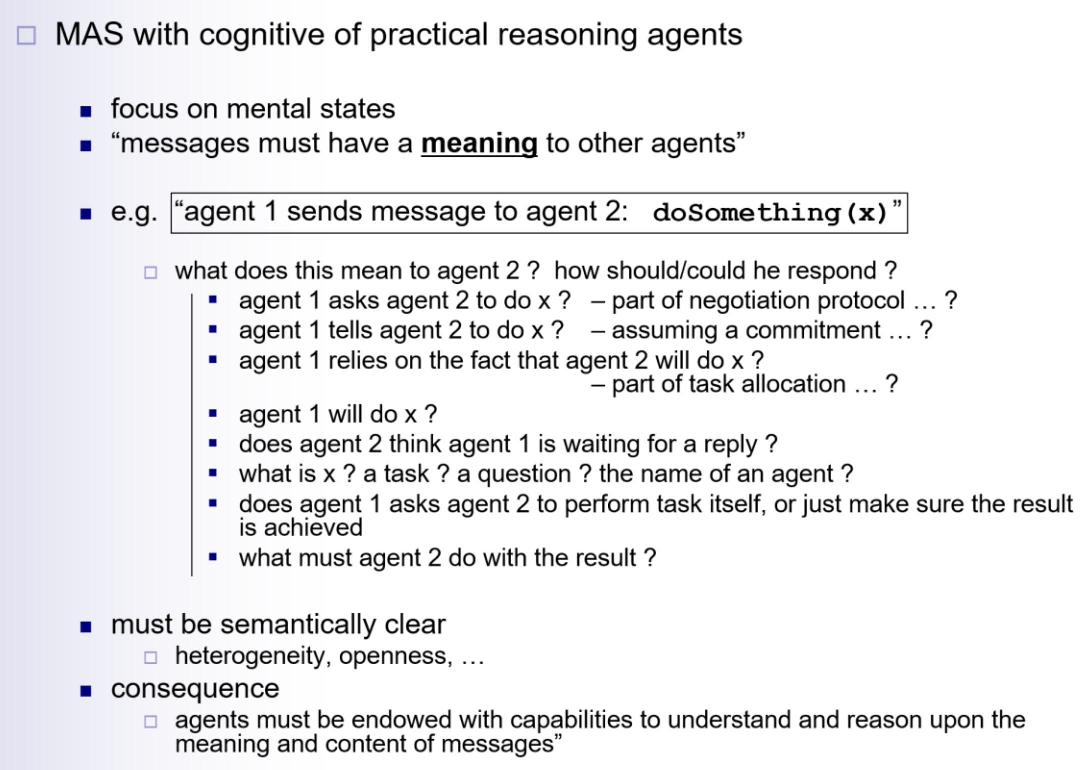
Agent communication

- message passing
  - some of the challenges
    - distributed systems
      - brokering, naming services, discovery, …
      - “infrastructure” for sending messages
      - heterogeneous entities
        - language, developer, execution environment, …
    - multi-agent systems



objects	→	messages
agents	→	**speech acts**

Agent Communication



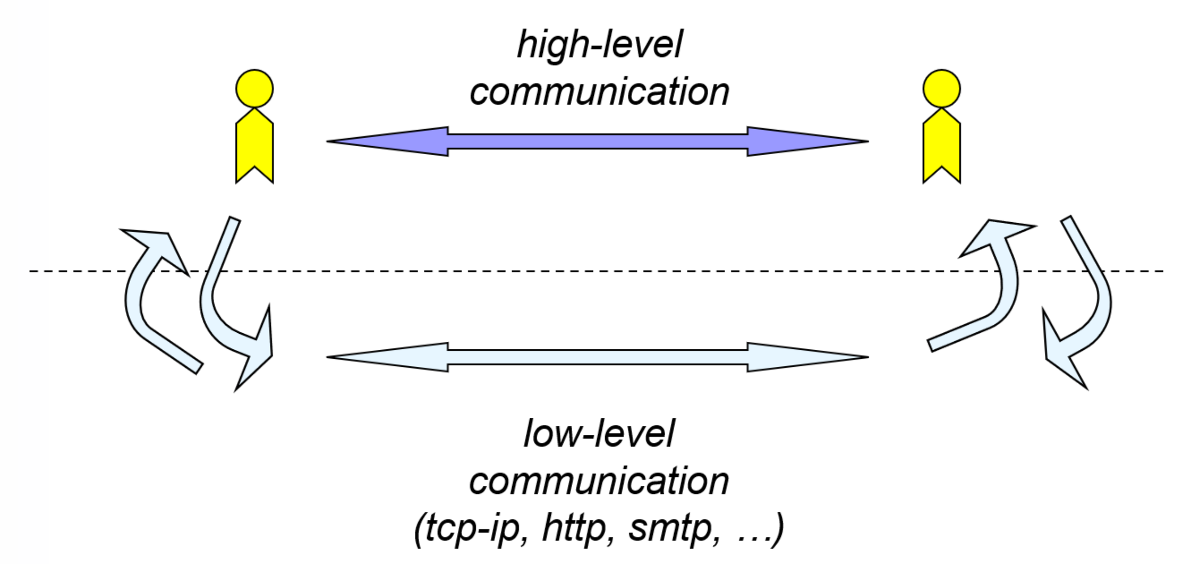
- MAS with cognitive of practical reasoning agents
  - focus on mental states
  - “messages must have a **meaning** to other agents”
  - e.g. “agent 1 sends message to agent 2: **`doSomething(x)`**”
    - what does this mean to agent 2 ? how should/could he respond ?
      - agent 1 asks agent 2 to do x ? – part of negotiation protocol … ?
      - agent 1 tells agent 2 to do x ? – assuming a commitment … ?
      - agent 1 relies on the fact that agent 2 will do x ? – part of task allocation … ?
      - agent 1 will do x ?
      - does agent 2 think agent 1 is waiting for a reply ?
      - what is x ? a task ? a question ? the name of an agent ?
      - does agent 1 asks agent 2 to perform task itself, or just make sure the result is achieved
      - what must agent 2 do with the result ?
  - must be semantically clear
    - heterogeneity, openness, …
  - consequence
    - agents must be endowed with capabilities to understand and reason upon the meaning and content of messages”

Agent Communication

- ability to exchange information requires
  - 1. ability to “physically” exchange information
  - 2. common understanding
  - 3. common language
  - 4. interaction strategies / protocols

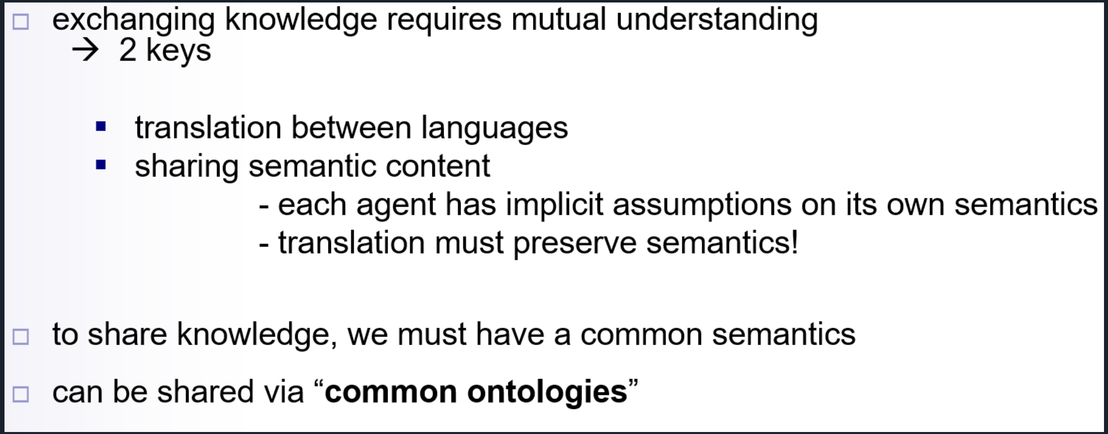
Agent Communication

- ability to exchange information requires
  - 1. ability to “physically” exchange information


high-level communication
low-level communication (tcp-ip, http, smtp, …)

Agent Communication

- ability to exchange information requires
  - 1. ability to “physically” exchange information
  - 2. common understanding



□ exchanging knowledge requires mutual understanding
→ 2 keys

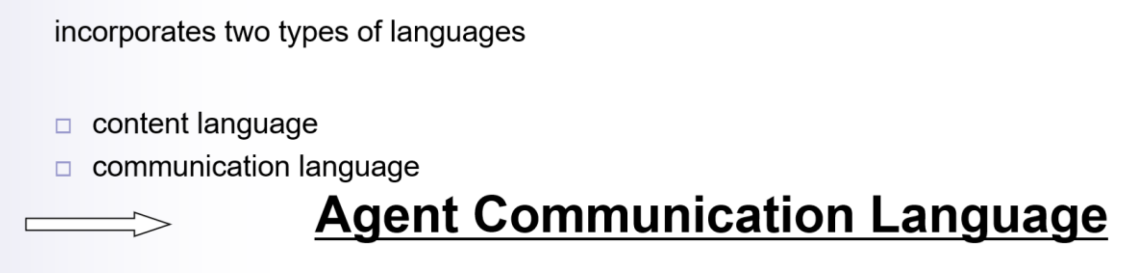
■ translation between languages
■ sharing semantic content
- each agent has implicit assumptions on its own semantics
- translation must preserve semantics!

□ to share knowledge, we must have a common semantics

□ can be shared via “**common ontologies**”

Agent Communication

- ability to exchange information requires
    1. ability to “physically” exchange information
    2. common understanding
    3. common language



incorporates two types of languages

□ content language

□ communication language

→ **Agent Communication Language**

Agent Communication Languages (ACL)

- Agents are typically defined at a “high” level
- an ACL should support intentional communication
  - the intentional descriptions use concepts such as: beliefs, goals, intentions, commitment

- the language should not define protocols such as

  - transport protocols
  - high level coordination protocols
  - constraints on valid exchanges

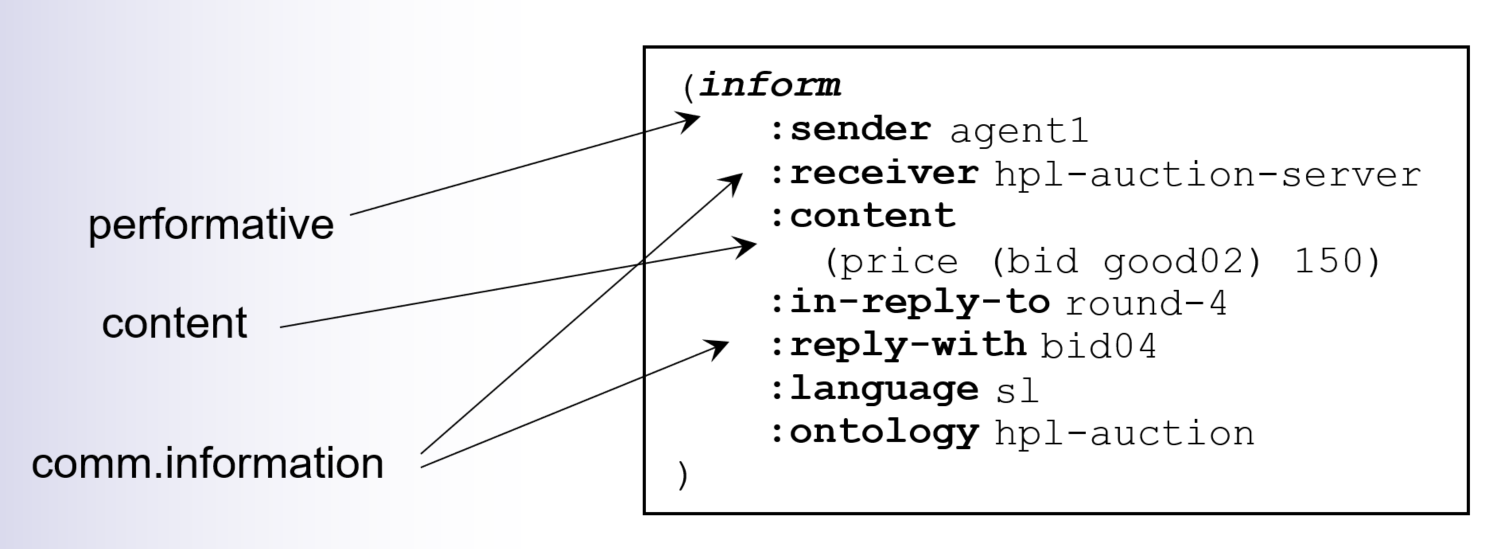
ACLs: FIPA ACL

- Foundation for Intelligent Physical Agents - http://www.fipa.org/
  - 1995
  - since 2005: IEEE Computer Society standards organization
  - “promotes agent-based technology and the interoperability of its standards with other technologies”

ACLs: FIPA ACL

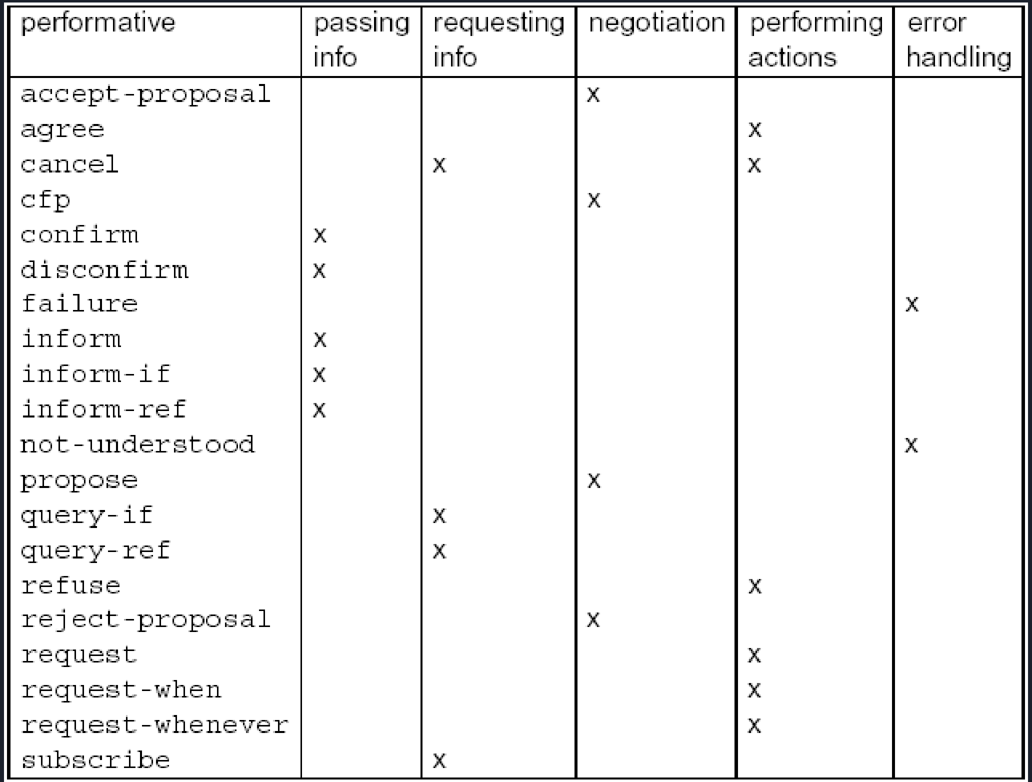
- standardisation of agent-related issues
    - FIPA-OS
    - FIPA infrastructure architecture
    - ...
    - FIPA-ACL
        - similar to KQML
        - consists of a set of message types and the description of their pragmatics — that is, the effects on the mental attitudes of the sender and receiver agents.
        - describes every communicative act with both a narrative form and a formal semantics based on modal logic.
        - separates the outer language (the intended meaning of the message) from the inner language (content language).

ACL Message



```
(inform
    :sender agent1
    :receiver hpl-auction-server
    :content
      (price (bid good02) 150)
    :in-reply-to round-4
    :reply-with bid04
    :language sl
    :ontology hpl-auction
)
```

FIPA ACL: performatives

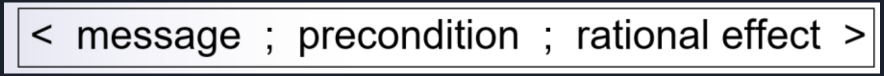


performative	passing info	requesting info	negotiation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x		x	
cfp			x		
confirm	x				
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				
not-understood					x
propose			x		
query-if		x			
query-ref		x			
refuse				x	
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

FIPA ACL: semantics in SL “the Semantic Language”

- SL (Semantic Language)
  - can represent propositions, objects, and actions

- formal semantics



< message ; precondition ; rational effect >

message	the content of the message
precondition	on the “situation” (mental state) of the sender
rational effect	intended effect on mental state of receiver

ACLs: KQML

- Knowledge Query Manipulation Language (KQML)
  - content - ignored by KQML messages
  - message
    - determines interaction types
    - supplies performative & content
    - may describe ontology, etc.
  - communication
    - low level communication parameters
    - sender, receiver, unique message ID

Negotiation protocols

Basic protocols:

- contract-net protocol
- auction protocols

Negotiation protocols

- … iterative communication among a group of agents in order to reach a mutually accepted agreement on something…..
- every day approach in resolving conflicts
- needed:
    - a set of options
    - a utility function
        - every option has a price and benefit
        - this function evaluates the worth of an option to an agent.
    - a negotiation protocol
        - multiple stages or steps in the negotiation process
        - eventually the process must either terminate or converge to a solution

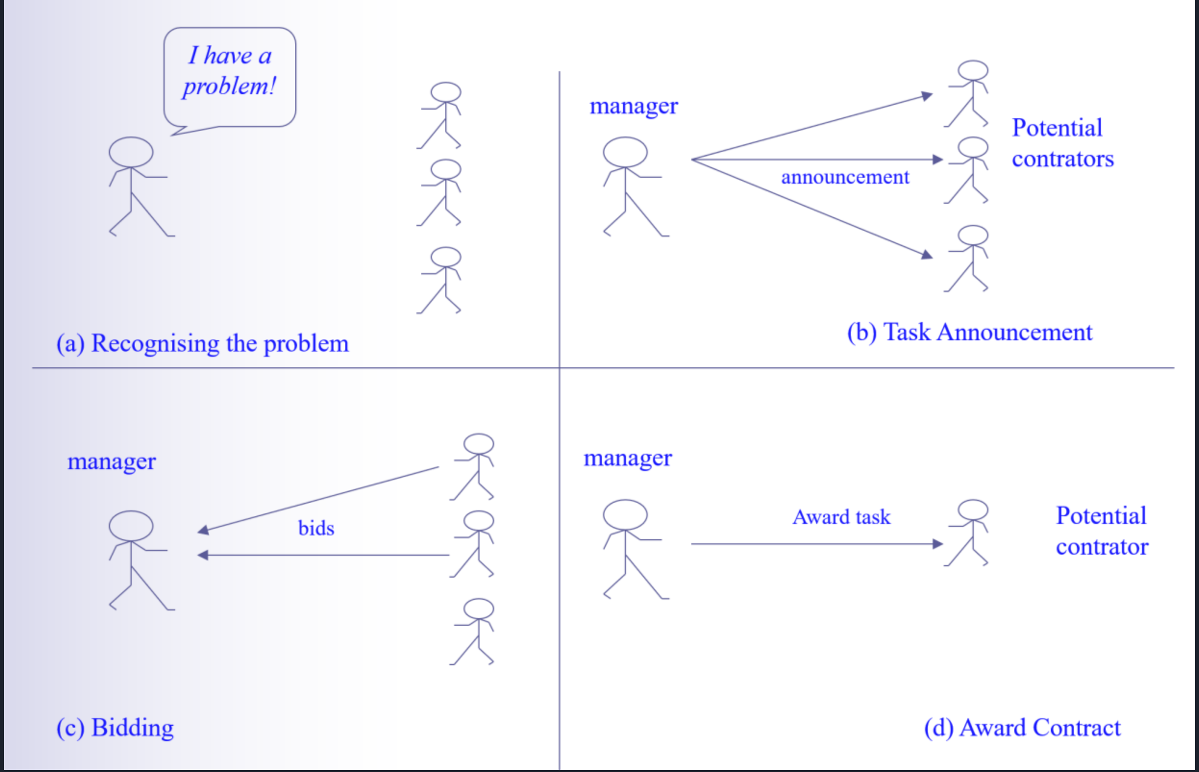
The Contract Net Protocol

- a manager
  - breaks the problem into several interacting sub-problems
  - looks for a contractor
  - selects the suitable contractor
  - assigns a sub-problem
  - monitors the progress of the overall solution
- a contractor
  - ‘bids’ for work
  - accepts a task
  - it has a binding agreement to complete the task according to the agreed terms and completes the task undertaken.
  - recursively, the contractor can be a manager for the task it has undertaken.

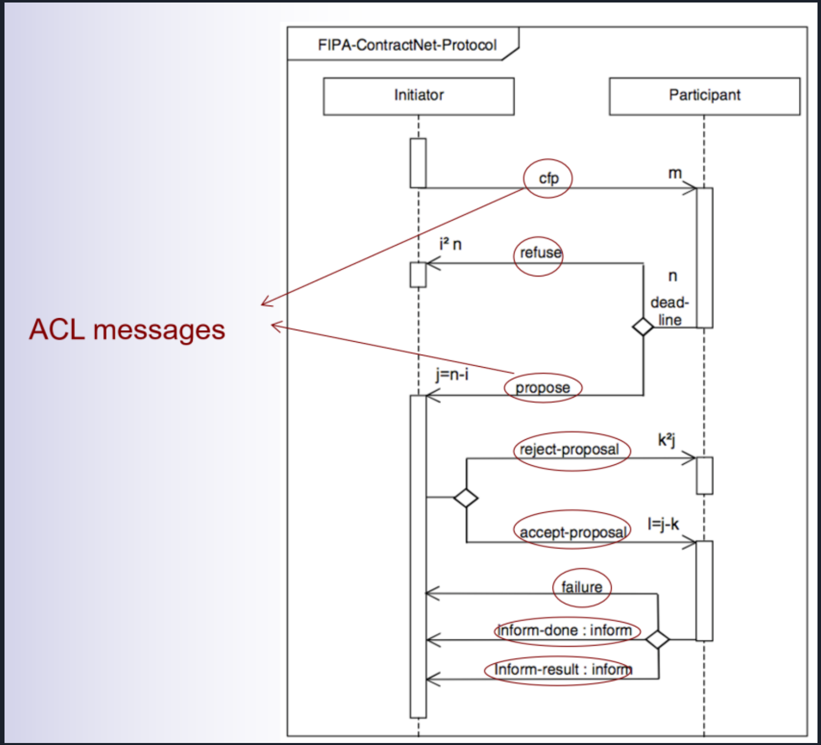
The Contract Net Protocol

- Basic assumptions:
  - the problem has a well defined structure for decomposing
  - coarse-grain decomposition is possible
  - there are enough contractors waiting to do the announced tasks

The Contract Net Protocol



The Contract Net Protocol



Applicability of Contract Net

The Contract Net is:

- a high-level communication protocol
- a way of distributing tasks dynamically
- decentralized / situated
- a means of self-organization for a group of agents

…but:

- limited (mostly for well-defined hierarchies of tasks)
- not scalable
- re-allocation ?

MAS as Distributed Systems

- Agents
  - Autonomous: independently acting
  - Heterogeneous: independently designed

- Agents communicate with each other
  - Protocols define how the agents ought to communicate with one another
    - A protocol is a modular, potentially reusable specification of the interactions between two or more entities
    - Defining a protocol helps ensure interoperability, i.e., being able to work together

Traditional Distributed Computing

- Ignore autonomy and heterogeneity
- Specify interaction in low-level operational terms via message order and occurrence
- Specify interoperation in low-level terms
- A system may be fragile because of its interoperation depending upon low-level details that can easily change when one of the parties modifies its internals

Autonomy

- Each agent is free to act as it pleases
  - We must design protocols so that they do not over-constrain an agent’s interactions
  - Intelligence is irrelevant in a protocol: must design a protocol whose correctness does not depend upon the agents’ internal reasoning

Autonomy

- The agents are the logical units of distribution
  - Physical distribution is based on considerations such as geographical distribution, throughput, redundancy
  - Cannot treat two or more agents as a single operating system process, even though that's how they may be realized, e.g., within the same virtual machine in an agent platform

Heterogeneity

- In traditional systems, it is enough that protocols specify the
  - Schemas of the messages exchanged
  - Legal flows, that is, their ordering and occurrence
- In multiagent systems, protocols must specify the meaning of the messages
  - Logically, agents interoperate on the basis of meanings of their communications
    - Since the meanings determine their social state, i.e., state of their interaction

Heterogeneity

- Whatever is in the protocol
  - Becomes the standard to which agents are implemented
  - Defines the level of heterogeneity: the agents can be heterogeneous with regard to everything else
  - Giving prominence to low-level concerns (such as ordering and occurrence of messages) couples the agent designs at the corresponding low level
    - Even though such concerns are appropriate for lower levels of the implementation

Distributed and Multiagent Systems

Distributed Systems and MAS:

- Similar concepts and concerns
- Similar objectives
- Similar problems: communication, coherence, results

Distributed and Multiagent Systems

- “Distributed” - refers to the system architecture
- “Multiagent” - refers to the problem solving method

When is a multiagent system also a distributed system?

Distributed and Multiagent Systems

Architectural organization

- Centralized
- Decentralized

Distributed and Multiagent Systems

Architectural styles

- Layered
- Object-based
- Resource-centered (Web: SOA)
- Event-based (Web: publisher-subscriber)

Distributed and Multiagent Systems

MAS-specific architectural properties

- Deliberative
- Reactive
- Hybrid

Distributed and Multiagent Systems

Distributed Systems Communication models

- Remote procedure calls
- Message-oriented communication
- Multicast communication

Distributed and Multiagent Systems

MAS-specific communication properties

- Agents engage in conversations (social aspect)
- Messages structured according to an Agent Communication Language (ACL)

Distributed and Multiagent Systems

Tools, technologies, frameworks

- MAS: specialized frameworks, protocols, languages
- Distributed Systems: modular frameworks and tools

Distributed and Multiagent Systems

When implementing a MAS:

- Specialized frameworks include ACLs and multiple agent-specific considerations
- Specialized agent knowledge is necessary
- Frameworks are not really modular

Distributed and Multiagent Systems

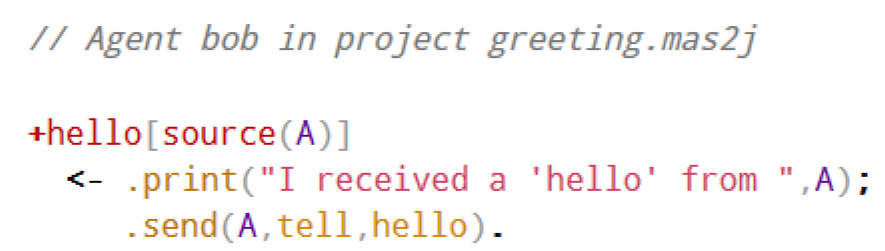
Case study: JaCaMo[http://jacamo.sourceforge.net/]

- Jason: an interpreter for AgentSpeak
- CArtAgO: a Java-based framework for environments in agent-oriented applications
- Moise: an organisational platform based on notions like roles, groups, and missions

Distributed and Multiagent Systems

Building a MAS with JaCaMo: Jason

- AgentSpeak: an agent programming language



```
// Agent bob in project greeting.mas2j

+hello[source(A)]
  <- .print("I received a 'hello' from ",A);
     .send(A,tell,hello).
```

Distributed and Multiagent Systems

Building a MAS with JaCaMo: CArtAgO

- An environment is composed of workspaces
- A workspace contains a basic set of predefined artifacts
- All agent’s actions are <u>determined</u> by the set of artifacts available/usable in the workspace

Distributed and Multiagent Systems

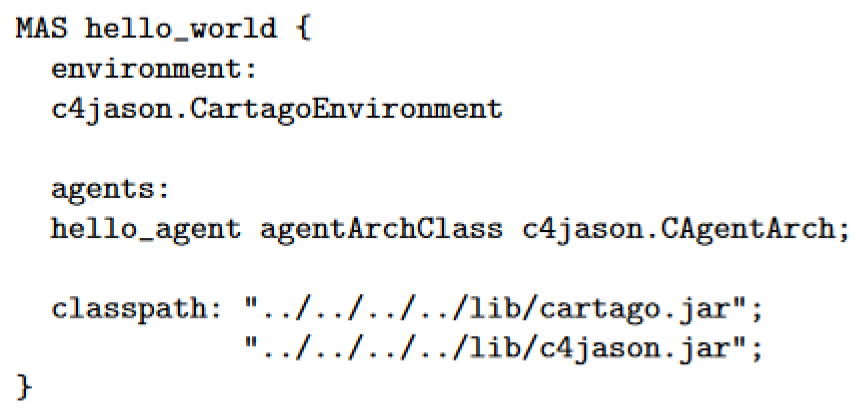
Building a MAS with JaCaMo: CArtAgO

- Communication between agents in the same workspace is handled internally (blackboard/RMI)
- Observable properties and events are mapped into beliefs
- Translation rules between Jason and CArtAgO

Distributed and Multiagent Systems

Building a MAS with JaCaMo: CArtAgO

- A domain-specific language is used



```
MAS hello_world {
  environment:
  c4jason.CartagoEnvironment

  agents:
  hello_agent agentArchClass c4jason.CAgentArch;

  classpath: "../../../../lib/cartago.jar";
             "../../../../lib/c4jason.jar";
}
```

Distributed and Multiagent Systems

Building a MAS with JaCaMo: Moise

- It enables an MAS to have an explicit specification of its organisation
- Structured in three levels: (i) individual agent tasks, (ii) agent structures and (iii) agent societies
- Uses the concepts of roles and missions

Distributed and Multiagent Systems

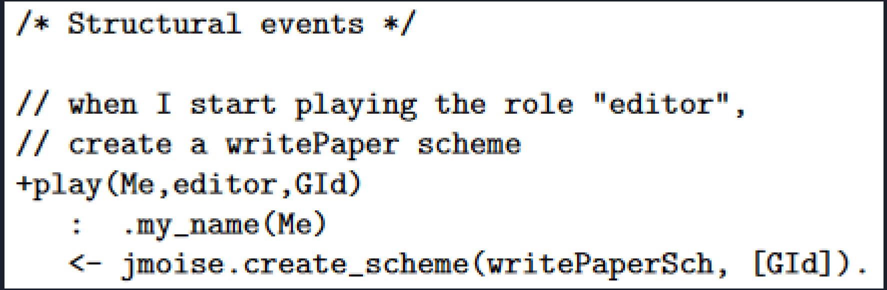
Building a MAS with JaCaMo: Moise

- Schemes (goals and plans) and missions follow a specific XML schema
- Normative specifications (also XML) states both the required roles for missions and missions obligations for roles

Distributed and Multiagent Systems

Building a MAS with JaCaMo: Moise

- Domain-specific language also in place



```
/* Structural events */

// when I start playing the role "editor",
// create a writePaper scheme
+play(Me,editor,GId)
   :  .my_name(Me)
   <- jmoise.create_scheme(writePaperSch, [GId]).
```

Distributed and Multiagent Systems

Building a MAS with other frameworks

- Similar restrictions and conditions
- Agent-specific capabilities and models are supported in a framework-by-framework basis
- Equivalent models and capabilities are not interoperable between frameworks (closed box)

Distributed and Multiagent Systems

Building a MAS with *any* framework, in general:

- Absence of agent standards leads to local models and implementations
- Multiple agent-oriented programming languages

Distributed and Multiagent Systems

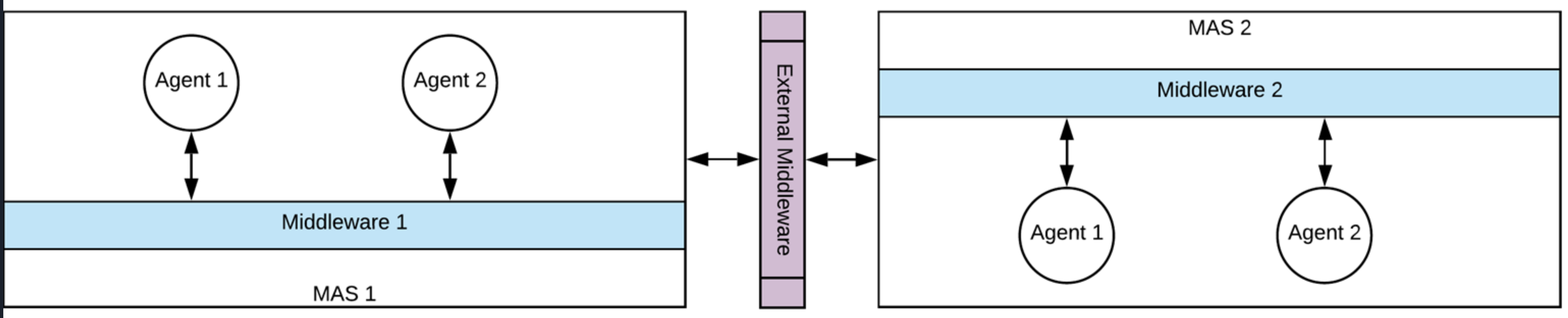
Building a MAS with *any* framework, in general:

- Agent-to-agent communication happens *within* the framework (blackboard, etc.)
    - Each framework has its own internal middleware

Distributed and Multiagent Systems

Building a MAS with *any* framework, in general:

- Communication MAS-to-MAS:



Programming MAS

Different MAS frameworks and tools

- https://mas-unige.github.io/fantastic_mass/frameworks.html

Programming MAS

Case study: SPADE

- Multi-agent platform based on XMPP
- Presence notification allows the system to know the current state of the agents in real-time
- Python >=3.8
- Asyncio-based
- Agent model based on behaviors
- Supports FIPA metadata using XMPP Data Forms (XEP-0004: Data Forms)
- Web-based interface
- Use any XMPP server

Practical Activity

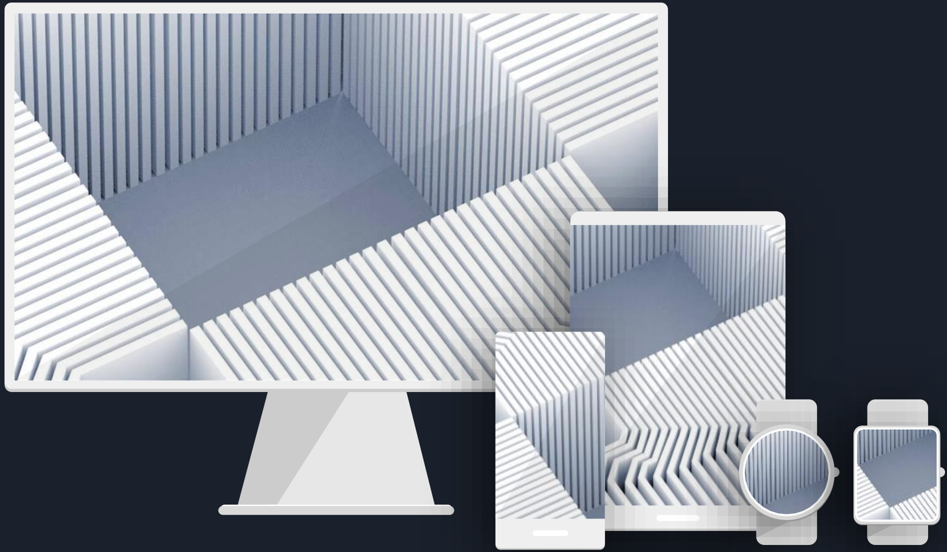
Getting started with MAS programming: SPADE

- Go to: https://spade-mas.readthedocs.io
- Install SPADE
- Read/Run: From “Quick Start” to “Extending SPADE with plugins”
- Create a message ring between 3 agents (see: e-disciplinas)
- Write a report on the task
- Deadline: 01/08/2023

Thank you!

anarosa.brandao@usp.br

arthur.casals@usp.br



References

1. Michael Wooldridge. An introduction to multiagent systems. Baffins Lane, John Wiley and Sons, 2009 2nd ed.
2. Gerhard Weiss (Ed). Multiagent systems. Cambridge, 2nd edition MIT Press, 2013.