Master of Technology in Knowledge Engineering

Unit 7: Developing Intelligent Systems for Performing Business Analytics

Distributed Intelligent Systems

Fan Zhenzhen
Institute of Systems Science
National University of Singapore
email: zhenzhen@nus.edu.sg

Objectives

On completing this module you will:

- Understand the main motivations for distributed intelligent systems and MAS
- Be aware of the micro and macro issues of developing multi-agent systems, a perspective of DIS
- Understand the main components in some agent architecture models
- Understand the basic concepts of the agent communication and interaction to achieve coordination

Agenda

- Distributed AI and MAS
- Micro Issues
- Macro Issues
- MAS Co-ordination
- Developing MAS: Tools & Methodology
- Summary

Distributed AI

- Distributed AI : distributed problem solving (DPS) and multi agents systems (MAS)
- DPS considers how a particular problem can be solved by a number of modules which cooperate in dividing and sharing knowledge about the problem and its evolving solutions.
- A MAS can be defined as a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver.

Distributed Problem Solving

- Multiple problem solvers (agents) combining their knowledge, information, and capabilities so as to develop solutions to problems that each could not have solved as well alone due to an inherent distribution of resources such as knowledge, capability, information, and expertise.
- It requires group coherence and competence to solve distributed problems well. The agents need to be cooperative and know how to work together.
- The teamed agents formulate solutions by each tackling (one or more) sub-problems and synthesizing these sub-problem solutions into overall solutions.

Characteristics of MAS

- Each agent has incomplete information, or capabilities for solving the problem, thus each agent has a limited viewpoint;
- There is no global system control;
- Data is decentralized; and
- Computation is asynchronous.

Motivations for Distributed IS

- To solve problems that are too large for a centralised single agent due to resource limitations or the sheer risk of having one centralised system;
- To provide solutions to inherently distributed problems, e.g. meeting scheduling or air-traffic control;
- To provide solutions which draw from distributed information sources, e.g. in domains of distributed sensor networks or distributed on-line information gathering, it is natural to adopt a distributed and collaborative agent approach
- To provide solutions where the expertise is distributed, e.g. in health care provisioning

Motivations for Distributed IS

- To enhance …
  - » modularity (which reduces complexity)
  - » speed (due to parallelism)
  - » reliability (due to redundancy)
  - » flexibility (i.e. new tasks are composed more easily from the more modular organization)
  - » reusability at the knowledge level (hence shareability of resources);

MAS Applications

- Industrial Applications
  - » Work flow and process management
  - » Manufacturing control
  - » Air Traffic Control
  - » Distributed sensing
- Medical Applications
  - » Patient Monitoring
  - » Health Care
- Commercial Applications
  - » Information retrieval and management
  - » Electronic Commerce
  - » Business Process Management
- Entertainment
  - » Interactive Theatre & Cinema
  - » Games

Issues in Building DIS

- How to formulate, describe, decompose and allocate problems, and synthesize results among a group of agents
- How to enable agents to communicate and interact
- How to ensure that agents act coherently in making decisions or taking actions, accommodating the global effects of local decisions and avoiding harmful interactions
- How to enable individual agents to represent and reason about the actions, plans, and knowledge of other agents in order to coordinate with them; how to reason about the state of the coordinated process (initiation and termination)
- How to recognize and reconcile disparate viewpoints and conflicting intentions

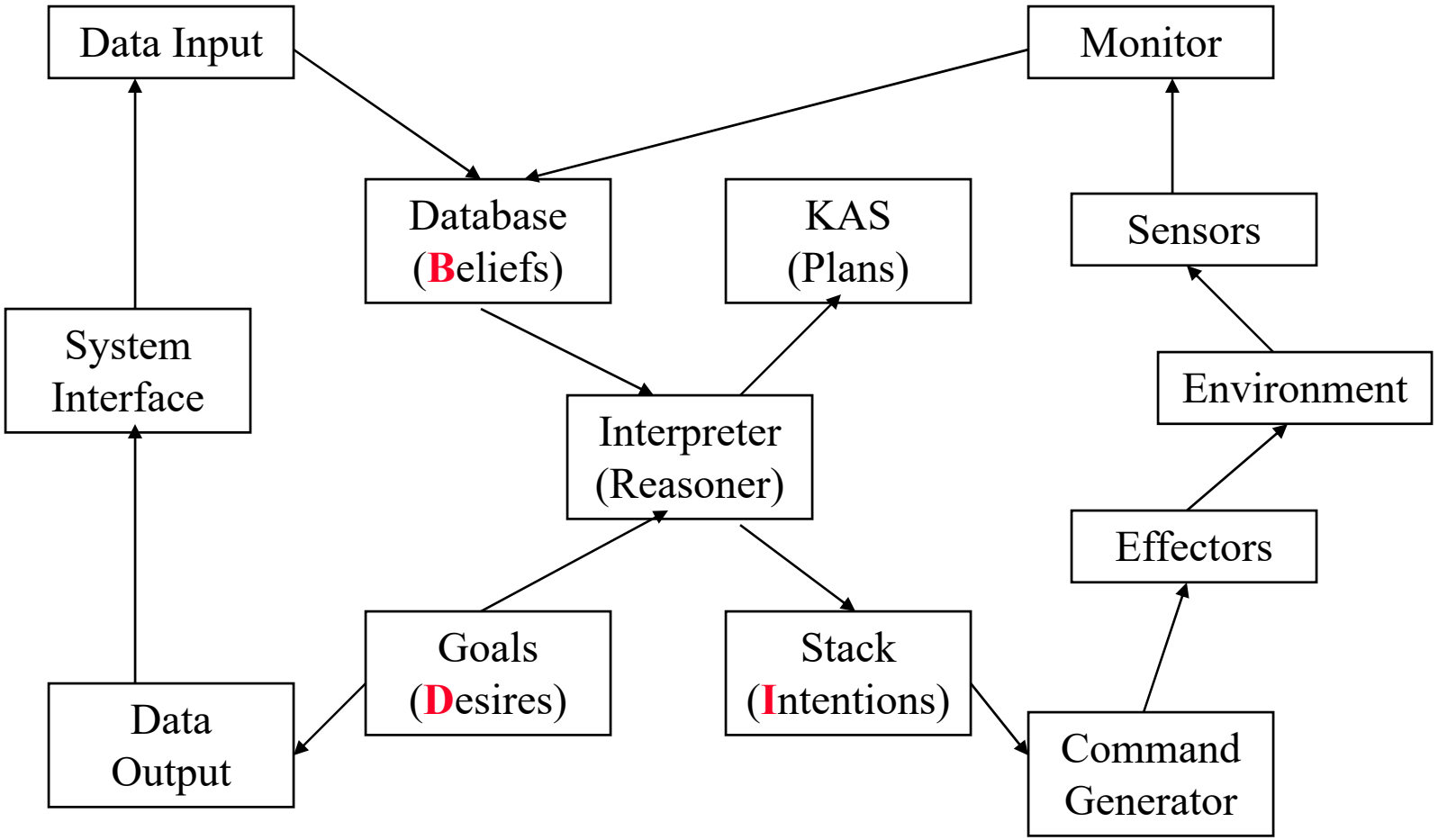
Micro Issues (Agent-Oriented)

- Building a system capable of flexible autonomous action is not easy!
- Reactive agents – behavioral, situated
- Deliberative agents – with reasoning ability
  - » Logical reasoning agents
  - » Practical reasoning agents – reasoning toward action
- Hybrid agents

Micro Issues

- Main problem: integrating reactive & goal-directed behaviour
- Many agent architectures, such as
  - » BDI architecture for deliberative agents
  - » Subsumption architecture for reactive agents
- Layered architectures (hybrid) currently popular
  - » Reactive layer
  - » Individual planning layer
  - » Social planning

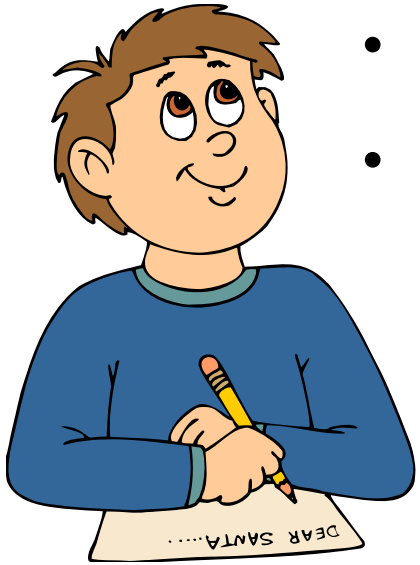
BDI Architecture



The BDI Model

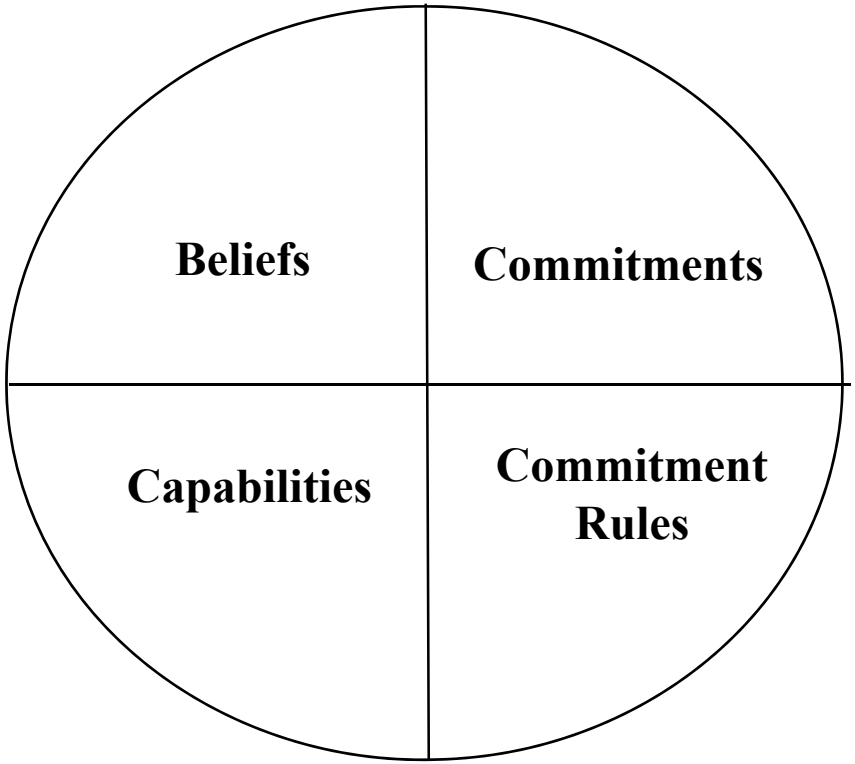
Agents using the BDI model have -

- Beliefs: Local knowledgebase
- Desires: What the agent is trying to achieve (similar to "goals")
- Intentions: Currently "adopted" plans
- Plans: Combinations of actions which achieve certain outcomes or respond to events & are used by the agent to further its intentions



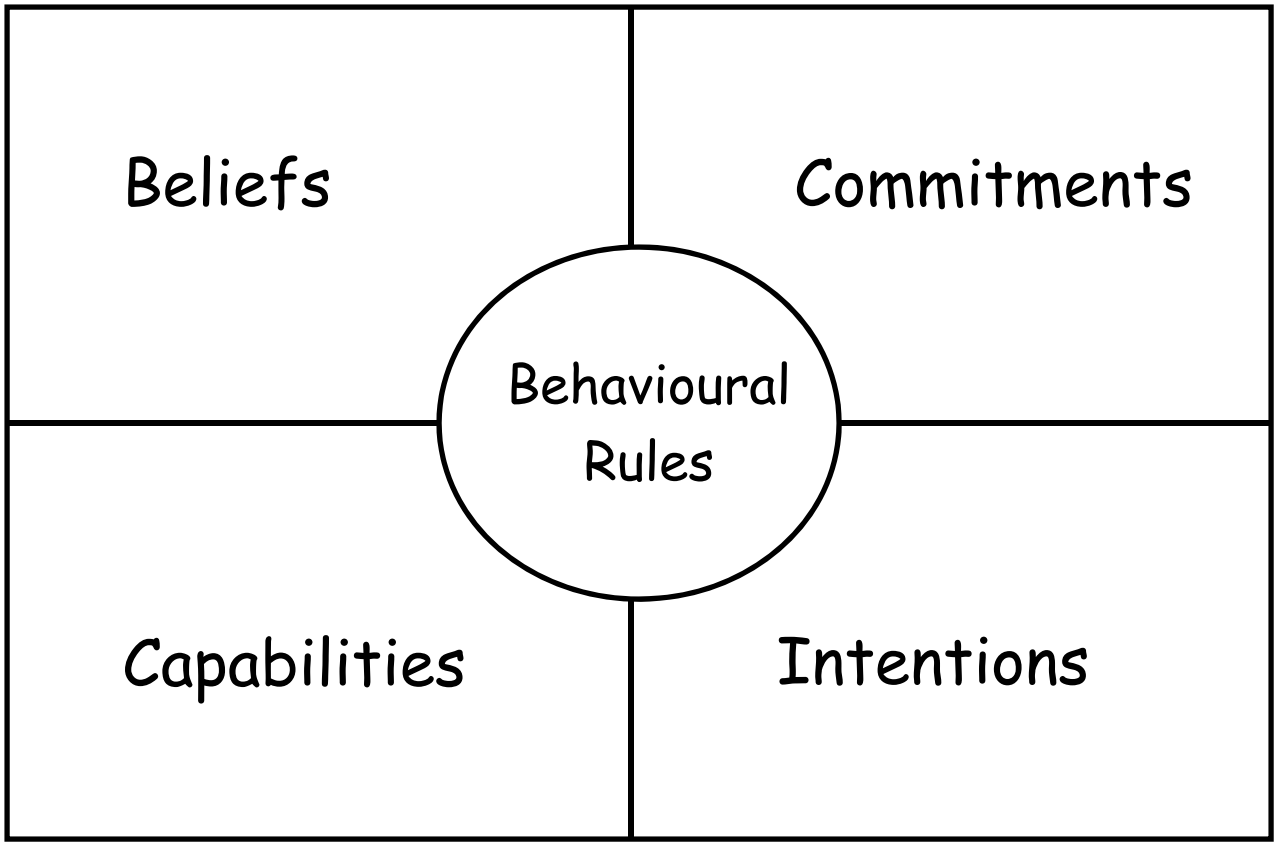
Shoham's Model

- Shoham (1993) theorized that cognitive agents possess a mental state which is composed of various mental elements.



Reticular Agent Mental Model

It extends Shoham's model



1. Beliefs

- Represent the current state of the agent's internal and external world
- They are updated as new information about the world is received
- Beliefs may be about:
  - » the world
  - » another agent's beliefs
  - » interaction (communication)with other agents
  - » its own beliefs

Beliefs for a Driver Agent

Belief Types

Location (Street, Street)
BlockedIntersection(Location)
Motion (Status, Direction, Speed)
Speed (Integer)
Status (String)
Street (String)
Direction (String)

Belief Instances

nextIntersection (Location)
currentLocation (Location)
destination (Location)
currentMotion (Motion)
trafficSpeed (Speed)

- First three are compositions
- The remaining are primitives

2. Capabilities

- Defines the actions the agent can perform provided the necessary preconditions are met
- Usually a list of capabilities is specified for an agent - actions that the agent can perform depend on the current mental state
- The capability is static and holds for the lifetime of the agent
- Actions may be:
  - » private
  - » communicative
- Format:

  Action: MakeQuery(database, patient, ?Query)

  Preconditions:database.Staus IS online

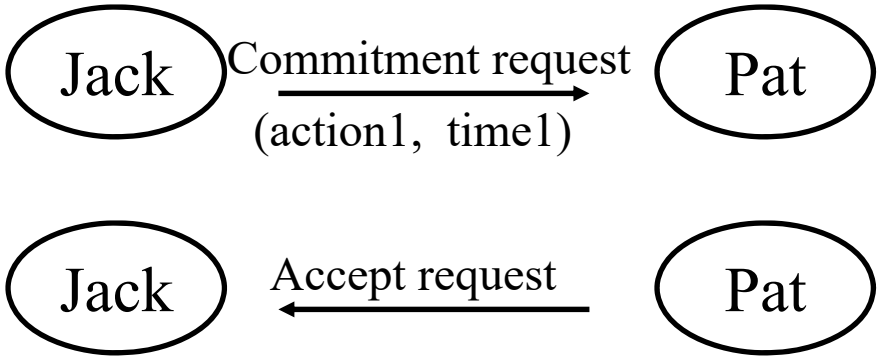
3. Commitments

- An agreement communicated to another agent to perform an action at a particular time

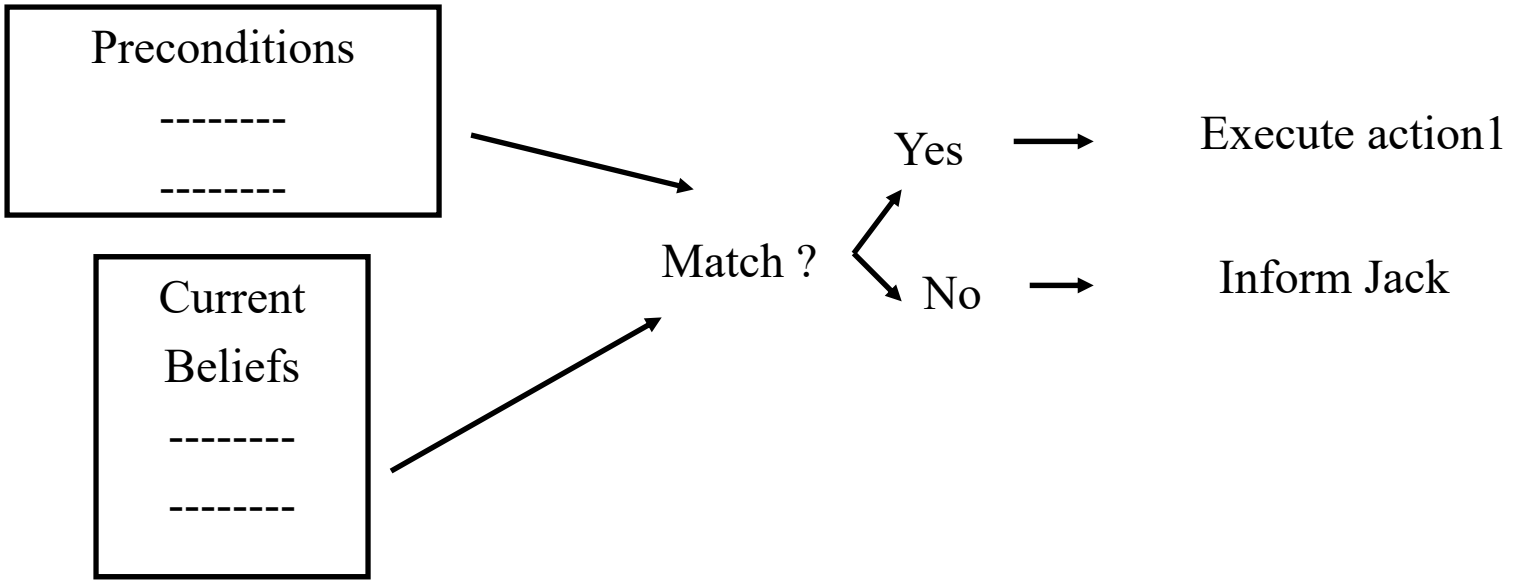
- An agent will not make a commitment if it does not have the capability to perform the action

- A commitment is not a guarantee that an action will be performed, it is only an " agreement to attempt an action if the preconditions are satisfied" - Why ?

Commitment Model



Pat: At time time1



4. Behaviour Rules

- **WHEN** (message conditions)
  - » addresses new events occurring in the environment
  - » may also include new messages received from other agents
- **IF** (mental conditions)
  - » compares the current mental model with the conditions that are to be satisfied for the rule
  - » patterns in the IF part are matched against the beliefs, commitments, capabilities and intentions in the mental model
- **THEN**
  - » defines the agent's actions and mental changes performed:
    - ♦ update of the mental model
    - ♦ communication to other agents
    - ♦ private actions internal to the agent

  (usually the action can include any combination of the above)

Behaviour Rule: Example

NAME "Redlight Stop Now Rule"

WHEN

?KQMLMessage.Performative EQUALS TELL

?KQMLMessage.Sender EQUALS "light-agent"

?KQMLMessage Content EQUALS String

?KQMLMessage.Ontology EQUALS "Trafficlights"

- **Used to test conditions of an incoming KQML message**
- **Evaluation continues to IF portion if all message conditions evaluate to true**
- **Note only the context type is checked here not the information itself**

Behaviour Rule: Example

IF

?KQMLMessage.Content EQUALS “light-red”

currentMotion.Status EQUALS moving

currentLocation EQUALS nextIntersection

- **Used to test conditions in the mental model**
- **The rule is activated only if all portions of the WHEN and IF are satisfied**

Behaviour Rule: Example

THEN

DO (Stop)

DO ($nextintersection = getNextIntersection(currentLocation, currentMotion.Direction))

ASSERT (SET_VALUE_OF currentMotion.Status TO stop)

ASSERT (SET_VALUE_OF nextIntersection to $nextIntersection)

SEND (performative =REPLY, receiver = "light-agent",
content = "acknowledged",
in-reply-to = ?KQMLMessage.Reply-with)

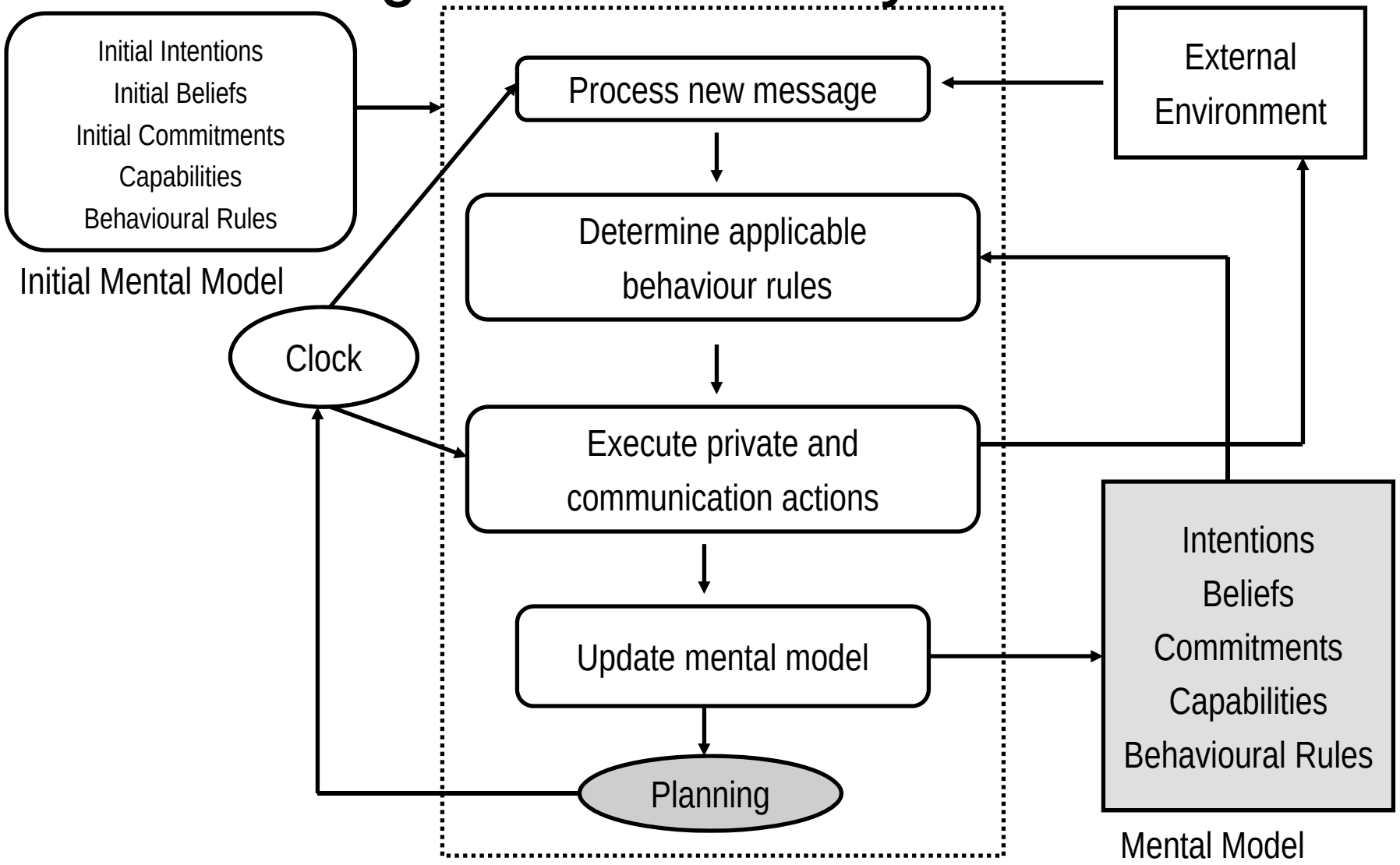
- **DO refers to private actions**
- **ASSERT or RETRACT refer to mental changes**
- **SEND refers to message transmission actions**

5. Intentions

- It is an agreement to achieve a particular state of the world/environment
- It is usually communicated to another agent

- How is it different from commitment ?
  - » commitment is an agreement to perform one action
  - » intention is an agreement to perform whatever actions needed to achieve the required state
- To satisfy intentions the agent will usually need to plan

Agent Execution Cycle



Macro Issues (MAS oriented)

- Next problem: how to get autonomous agents to cooperate effectively
- Two major issues
  - » Getting agents to talk to each other
    - ♦ ACL
  - » Getting agents to cooperate/negotiate
    - ♦ Cooperation protocol
- Must deal with possibility of conflict

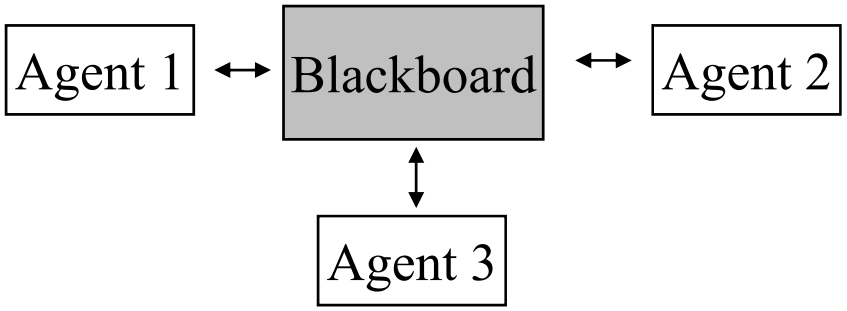
Architectures for communication

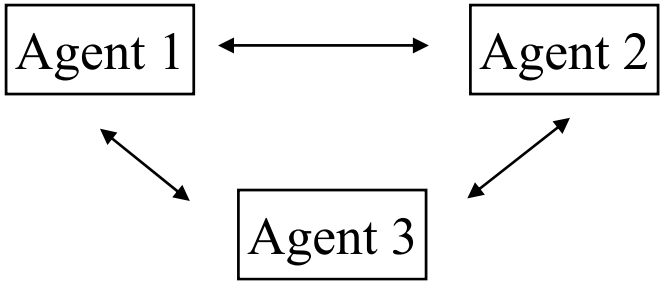
Hub-and-spoke

- Blackboard architecture
- Each agent sends the information to the blackboard (server)
- The information is available to all agents
- No direct communication between agents
- Simple architecture

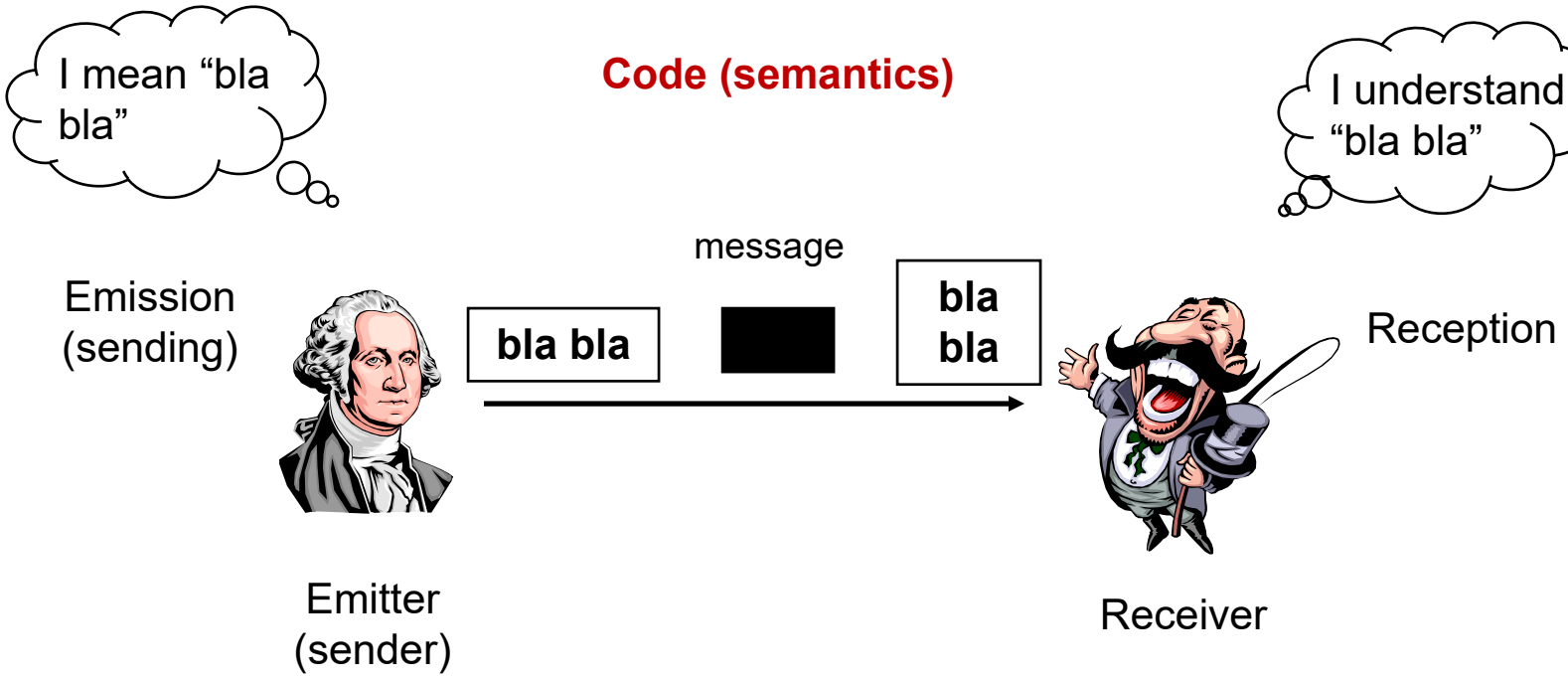
Peer-to-Peer

- No central server
- The information is available to the receiver agent
- Direct communication between the agents
- More complex architecture compared to hub-and-spoke





Classical Model of Communication



Communication Stack

Sequence of communicative acts *(Getting information about a CD and buying it)*	Conversation *(ACL Interaction Protocols, etc.)*
Communicating about a piece of content *(requesting to perform the action of buying a CD)*	Communicative Act *(ACL and KQML Performatives, etc.)*
Description of the states of the objects *(expressing the action of buying a CD)*	Content Expression *(KIF, SL, LOOM, etc.)*
Description of objects *(Meaning of "person" and "age")*	Ontology *(OOHVR, ebXML, etc.)*
Representation of content	Syntax *(XML, SOAP, HTML, etc.)*
Message Transfer *(data exchange protocols)*	Network Protocol *(HTTP, IIOP, WAP, etc.)*
Data Transfer *(low-level protocols)*	Transport *(TCP/IP, UDP, etc.)*

Speech Act Theory

- Concept developed initially in the context of the philosophy of language
- Communicating is acting: Sentences are not only true or false, they perform speech actions
- Classification of Speech Acts:
  - » **Assertive**: gives an information about the world by asserting something
  - » **Directive**: gives directives for the interlocutor
  - » **Promissive**: engages the locutor to accomplish certain acts in the future
  - » **Declarative**: accomplishes an act with the very pronouncement of the statement
  - » **Expressive**: gives the interlocutor indications about the mental state of the locutor
- Well-known interaction languages include KQML and ACL

Knowledge Query & Manipulation Language (KQML)

- Part of the Knowledge Sharing Effort of Defense Advanced Research Projects Agency (DARPA), which is aimed at developing techniques and methodology for building large-scale knowledge bases which are sharable and reusable.
- KQML is a high-level, message-oriented, communication language and protocol for exchanging information independently of content syntax and ontology
- KQML is independent of:
  - » The transport mechanism (e.g., http, tcp/ip, smtp, IIOP etc.)
  - » The content language (KIF, FIPA-SL, FIPA-CCL, etc.)
  - » The ontology assumed by the content
- KQML includes primitive message types to build agent architectures of various nature (e.g., for mediators, sharing intentions, etc.)

KQML: Examples

(*ask*

: *sender* john-agent

: *receiver* cheap-travel-server

: *reply-with* john-quote

: *language* loom

: *ontology* air-travel

: *content* (COST (ROUND-WORLD ?airline ?price)))

(*tell*

: *sender* cheap-travel-server

: *receiver* john-agent

: *in-reply-to* john-quote

: *language* LPAROLOG

: *ontology* air-travel

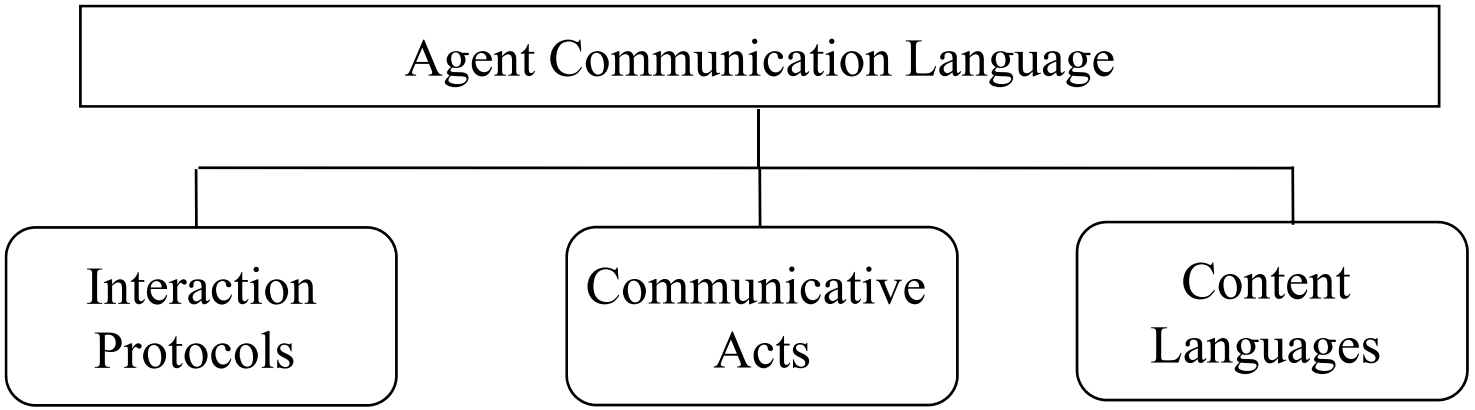
: *content* (COST (ROUND-WORLD, ba, £1750)))

- Defines a single speech act *ask* or *tell*, along with its attributes
- Communication layer: value of the keywords :*reply-with*, :*sender*, :*receiver*
- Message layer: performative name *ask*, :*language* and :*ontology*
- Content layer: value of the :*content* keyword

KQML Performatives

Category	Performatives
Basic informational	tell, deny,untell, cancel
Basic query	evaluate, reply, ask-if, ask-about, ask-one, ask-all, sorry
Mutli-response query	stream-about, stream-all
Basiceffector	Achieve,unachieve
Generator	standby, ready, next, rest discard, generator
Capability definition	advertise
Notification	subscribe, monitor
Networking	Register,unregister, forward, broadcast, pipe , break
Facilitation	broker-one, recommend-one, etc.

FIPA: Agent Communication Language (ACL)



- FIPA-ACL is designed to work with any content language and ontology
- FIPA-ACL syntax is quite similar to KQML except for different names for some communicative acts (22 acts)

FIPA ACL: Message Parameters

Parameter	Category of Parameters
Performative (mandatory)	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

FIPA CA: Assertive & Asking

- ***inform***: the sender informs the receiver that a given proposition is true
- ***confirm***: the sender informs the receiver that a given proposition is true, where the receiver is known to be uncertain about the proposition
- ***disconfirm***: the sender informs the receiver that a given proposition is false, where the receiver is known to believe, or believe it likely that, the proposition is true
- ***query-if***: the action of asking another agent whether or not a given proposition is true.
- ***query-ref***: the action of asking another agent for the object referred to by a referential expression
- ***subscribe***: the act of requesting a *persistent* intention to notify the sender of the value of a reference, and to notify again *whenever the object identified by the reference changes*
- ***not-understood***: the sender A informs the receiver B that it perceived that B performed some action, but A did not understand what B did

FIPA CA: Directives & Responses

- ***request***: the sender requests the receiver to perform some action
- ***request-when***:
  The sender wants the receiver to perform some action *when some given proposition becomes true*.
- ***request-whenever***:
  The sender wants the receiver to perform some action *as soon as some proposition is true and thereafter each time the proposition becomes true again*.
- ***agree***: the action of agreeing to perform some action, possibly in the future
- ***refuse***: the action of refusing to perform a given action and explaining the reason for the refusal.
- ***failure***: the action of telling another agent that an action was attempted but the attempt failed.
- ***cancel***: the action of one agent informing another agent that the first agent no longer has the intention that the second agent perform some action.

FIPA CA: Negotiation & Brokering

- ***cfp***: the action of calling for proposals *to perform a given action*
- ***propose***: the action of submitting a proposal to perform a certain action, given certain preconditions
- ***accept-proposal***: the action of accepting a previously submitted proposal to perform an action
- ***reject-proposal***: the action of rejecting a proposal to perform some action during a negotiation
- ***propagate***:
  The sender intends that the receiver
  - » treats the embedded message as sent directly to it (do as told)
  - » identify the agents denoted by the given descriptor
  - » send the received propagate message to them (message forwarding)
- ***proxy***:
  The sender wants the receiver to
  - » select target agents denoted by a given description
  - » send an embedded message to them (message forwarding)

FIPA CA – Macro Acts

Communicative Acts

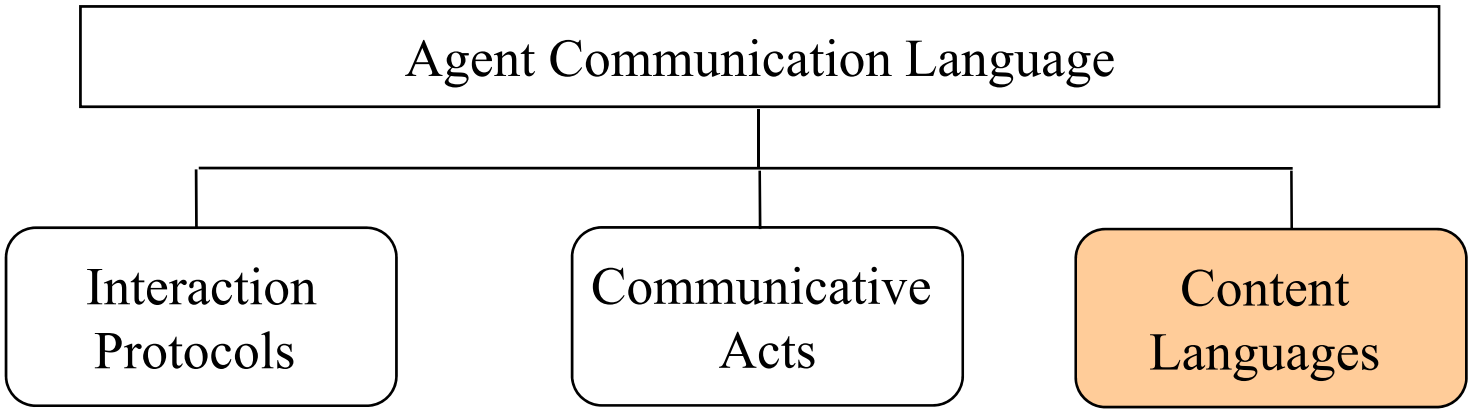
- ✓ Can be directly performed
- ✓ Can be planned by an agent
- ✓ Can be requested of one agent by another

Macro Acts

- ✗ Can be directly performed
- ✓ Can be planned by an agent
- ✓ Can be requested of one agent by another

- ***inform-if***: a macro-action for the agent of the action to inform the recipient whether or not a proposition is true.
- ***inform-ref***: a macro-action for the sender to inform the receiver the object which corresponds to a descriptor, for example a name

FIPA Content Languages



- Specifications deal with different representations of the content of ACL messages.
- Specifications
  - **FIPA-SL (Semantic Language) Content Language**
  - FIPA-CCL (Constraint Choice Language) Content Language
  - FIPA-KIF (Knowledge Interchange Format) Content Language
  - FIPA-RDF (Resource Description Framework) Content Language

FIPA: Semantic Language (SL)

Three types of content expressions

Examples

- **Proposition**
  - » A well formed formula in predicate calculus

(Works-for Zhenzhen ISS)
(fruit apple)

- **Action expression**
  - » A single action or a composite action built using the sequencing and alternative operators, introduced by keyword *action* and the actor(agent)

(action (agent-identifier :name A)
(Sell (Book :title "Integration")))

- **Identifying reference expression (IRE)**
  - » Uses referential operators such as ***iota*** to reference variable substitutions that would make a certain proposition true

(iota ?x (Works-for ?x ISS))

Communicative Act and Content

Activity:

- Fill the most suitable communicative acts for the given content, from INFORM, QUERY-IF, REQUEST, QUERY-REF

Communicative Act	Content
	(Works-for Zhenzhen ISS)
	(action (agent-identifier :name A) **(Sell (Book :title "Integration")))**
	(iota ?x (Works-for ?x ISS))

Example of using *iota*

- an interaction between agent A and B
- agent A is supposed to have the following knowledge base KB={School(ISS), Works-for(Zhenzhen, ISS), Works-for(Fangming, ISS)}.

```
(query-ref
  :sender (agent-identifier :name B)
  :receiver (set (agent-identifier :name A))
  :content
          "((iota ?x (Works-for Zhenzhen?x)))"
  :language fipa-sl
  :reply-with query2)
```

```
(inform
  :sender (agent-identifier :name A)
  :receiver (set (agent-identifier :name B))
  :content
      "((= (iota ?x (Works-for Zhenzhen ?x)) ISS))"
  :language fipa-sl
  :in-reply-to query2)
```

- a failure occurs if *no* object or *more than one* object satisfies the condition specified in the iota operator.

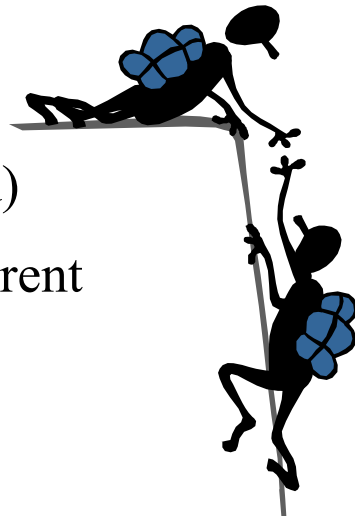
FIPA-SL: other Referential Operators

- ***Any*** – to denote any object that satisfies the proposition
  - » Failures only occur if there are no objects satisfying the condition.

- All – to denote the set of all objects that satisfy the proposition
  - » When no objects satisfy the condition, the identifying expression denotes an empty represented as "(set)".

```
(inform
  :sender (agent-identifier :name A)
  :receiver (set (agent-identifier :name B))
  :content
      "((= (all ?x (Works-for ?x ISS)) (set Zhenzhen Fangming))"
  :language fipa-sl
  :in-reply-to query2)
```

Co-ordination



- Managing inter-dependencies between the activities of agents
- An agent reasoning about its local actions and the (anticipated) actions of others to try to ensure the community acts in a coherent manner
- Representing **synchronising** and **conflict avoidance**
- Co-ordination is "the additional information processing performed when multiple, connected agents pursue goals that a single agent pursuing the same goals would not perform" (Malone, 1988)
- The purpose of co-ordination is to increase effectiveness (performance and survival) when the number of agents (and resources) increases

Coordination through Planning

Find a sequence of operators. Each operator is taken as a transition in a state space. The solution is obtained by finding a Path from the initial state to the final state in the search space.

- Centralised planning for multiple agents
- Centralised co-ordination for partial plans
- Distributed co-ordination for partial plans

Conflict Resolution

Agents pursuing similar or different goals will have to face conflicts

- Resource accessibility
- Alternative solutions
- Conflicting interests or goals

Examples of conflict resolution techniques

» Flip a coin

» A priori solution using strength, authority, …

» Mediation by a third agent that knows about the different points of view, and tries to solve the conflict

» Negotiating agents

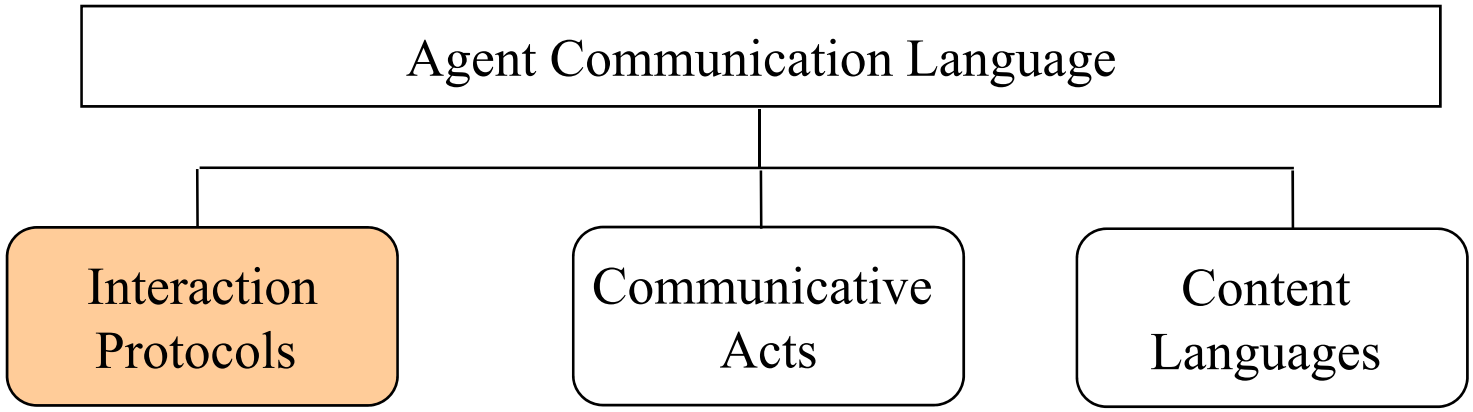
Negotiation Agents

Enter a transactional phase (exchanges, compromises, persuasive arguments, disagreement with the compromise or argument, requests for additional information, reasons for disagreement, utilities / preferences for the disagreed-upon issues) in order to reach an agreement, i.e., an equilibrium state

- Step 1: A proposes a solution
- Step 2: B evaluates the solution, determines its satisfaction
- Step 3: If B is satisfied, OK; Otherwise B proposes another solution with regards to its own goals & constraints
- Step 4: Goto step 1, exchanging the roles of A & B

Coordination with Interaction Protocol

- Coordination can also be achieved by developing agents following pre-agreed message exchange protocols ("social laws")



- **Specifications**

Request Interaction Protocol

Query Interaction Protocol

Request When Interaction Protocol

Contract Net Interaction Protocol

Iterated Contract Net Interaction Protocol

English Auction Interaction Protocol

Dutch Auction Interaction Protocol

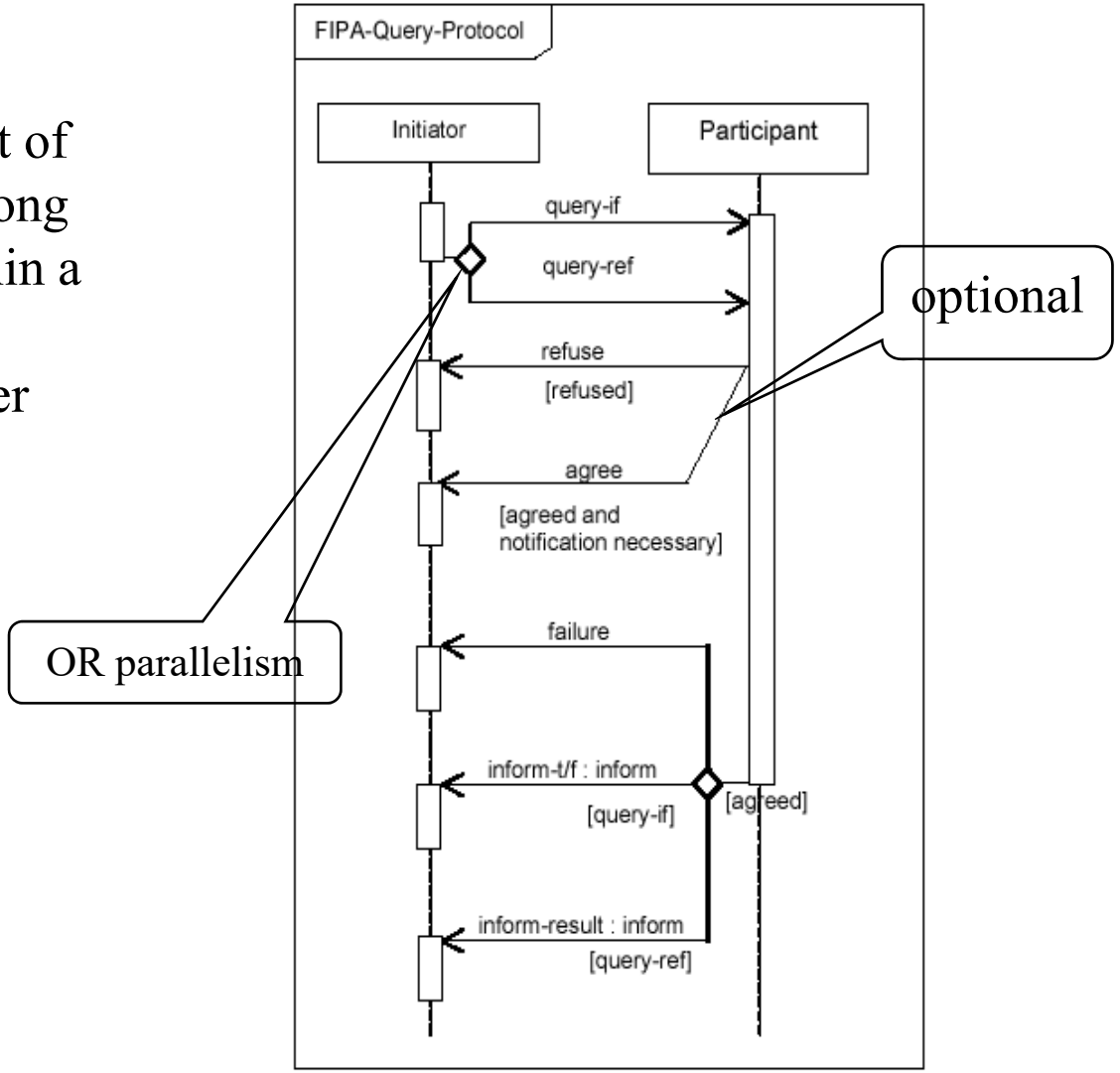
Brokering Interaction Protocol

Recruiting Interaction Protocol

Propose Interaction Protocol

FIPA Query Interaction Protocol

- Protocol Diagrams: an interaction, which is a set of messages exchanged among different agent roles within a collaboration to effect a desired behaviour of other agent instances.
- Vertical dimension: time
- Horizontal dimension: AgentRoles
- Conversation-id



Contract Net Protocol

1. Agent *manager* with a task decomposes it into subtasks to be contracted and announces the subtask
2. *Contractor* agents that are eligible to bid can submit bids
3. After enough time has elapsed, *manager* chooses from among submitted bids and makes one or more awards
4. Winning *contractors* provide interim and final reports of subtask accomplishment

- Contractors however can also become a manager who divides the problem again and contract it out to other contractors.

Decompose Task

- **Decompose task**
  - » Agent with a task decomposes it into subtasks to be contracted:
  - » Not very simple ! Knowledge of how to decompose (and recompose) must be available to the manager
  - » Often, there are alternative decompositions. Try them at the same time? Or pick the most promising one? How?
- **Announce task**
  - » That *manager* announces the subtask which includes
  1. Eligibility specifications - Constraints on who is allowed to bid
     - ♦ Balance can be difficult: too open means too many bids (and wasted resources); too closed might mean no acceptable bids, so have to retry
  2. Bid specifications - What should the bid contain
  - » How long should the manager wait until bid is closed?
  - » What happens when some contractors have not bid?

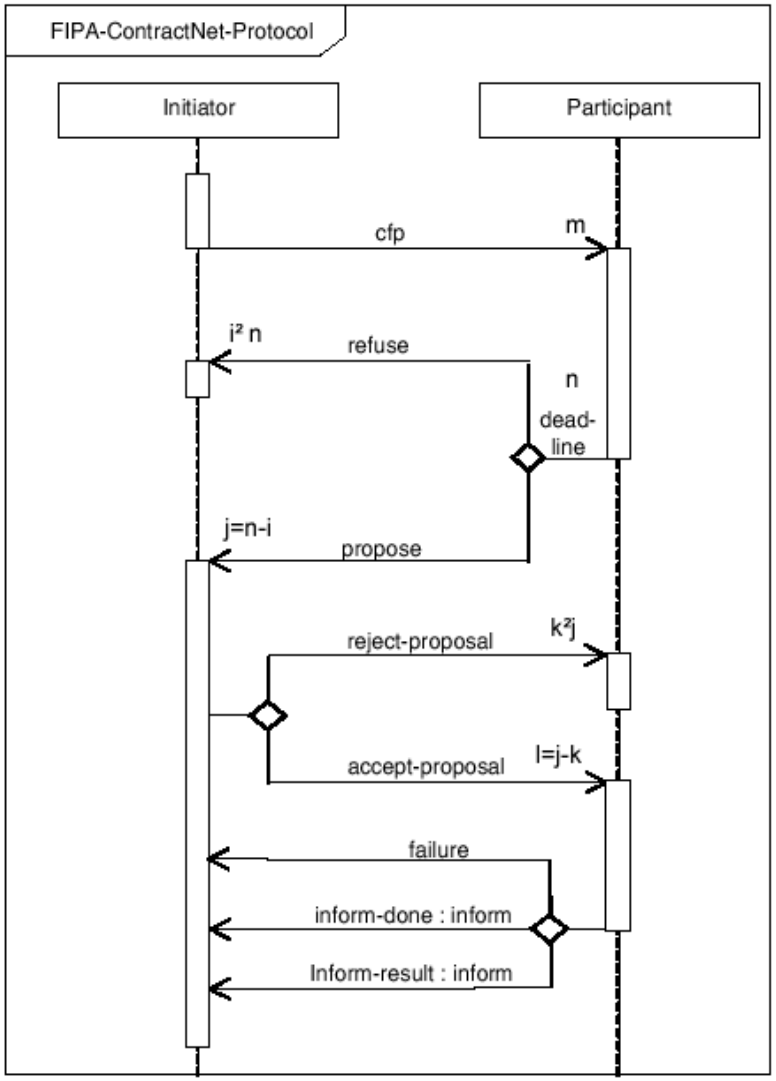
Submit Bids

- **Submit bids**
  - » *Contractor* agents that are eligible to bid can submit bids
  - » A "bid" does not necessarily involve (monetary) compensation for services
  - » A bid often specifies how well the task can be completed
    - ♦ Satisfying requirements
    - ♦ Timeliness
- **Select and award bids**
  - » Manager chooses from among submitted bids and makes one or more awards
  - » How the manager chooses is application dependent
    - ♦ If no degrees of "how well" then choose randomly (or lowest cost)
    - ♦ If degrees of "how well" then weigh the various factors

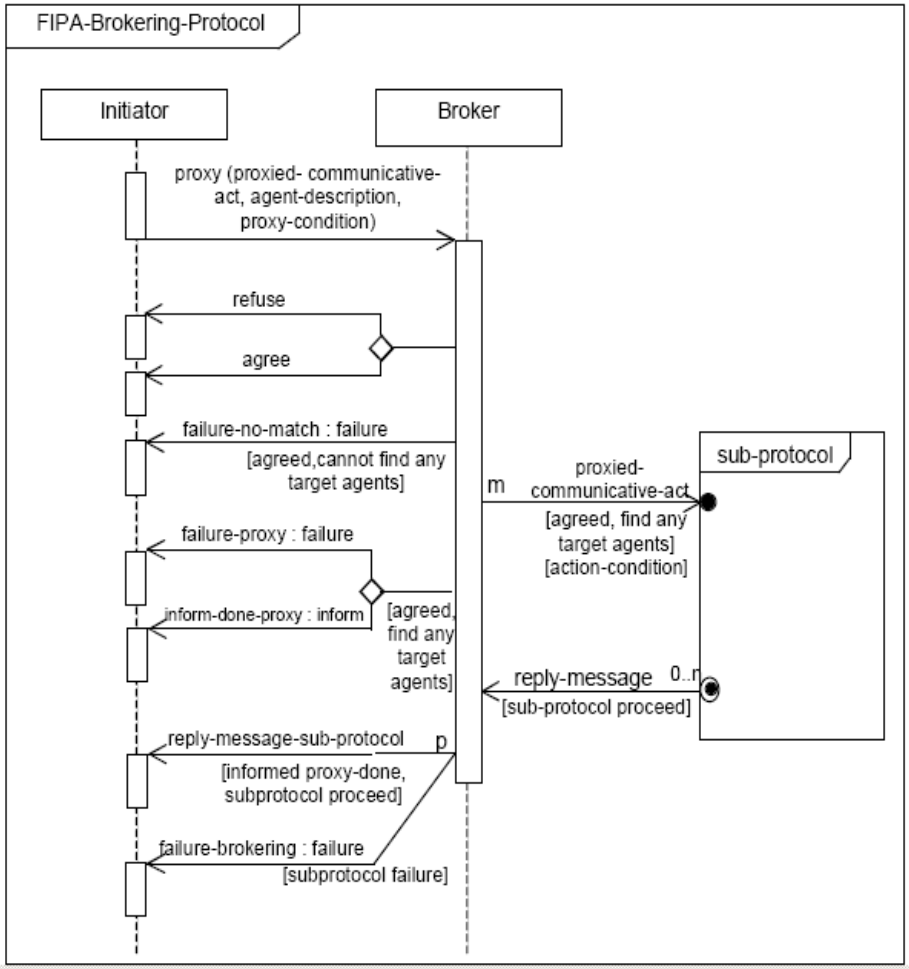
Periodic Reports

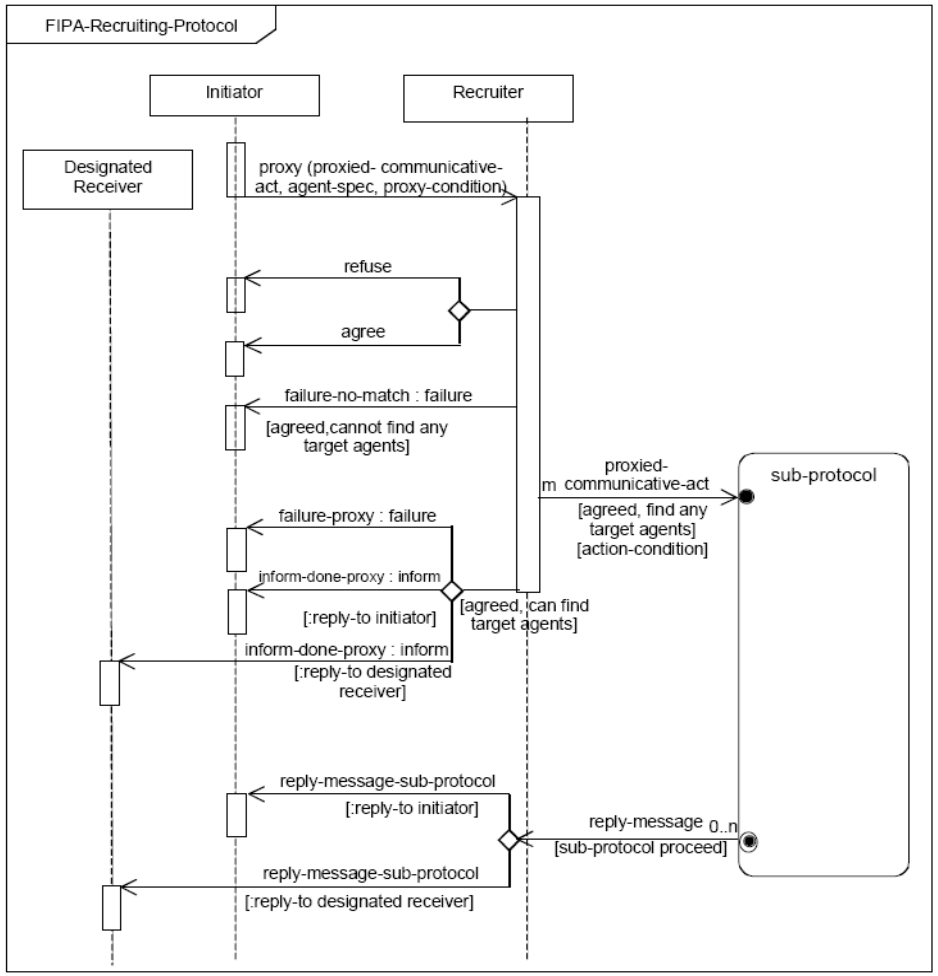
- Winning contractor(s) provide(s) interim and final reports of subtask accomplishment
- Interim reports can serve as reassurance to the manager that subtask is active
- Interim reports can help manager initiate or redirect activities of other contractors
- Final report provides subtask result to be synthesized into a complete task result

Contract Net Protocol



Brokering & Recruiting Protocols





Coordination with Organisations

In the context of Computer Science:

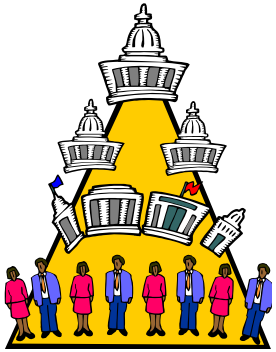
"An organisation is a decision & communication schema which is applied to a set of actors that together fulfill a set of tasks in order to satisfying goals while guaranteeing a global coherent state."

- An organisation provides a framework for agent interactions through the definition of roles, behavior expectations, and authority relations.
- The organizational structure imposes constraints on the ways the agents communicate and coordinate.

MAS Organisations

Classifications:

- Unstructured: Simple Groups, Teams, Special Interest Groups, Communities of Practice, …
- Centralised: Simple Hierarchies, Multi-level Hierarchies, Recursive Structures, …
- Decentralised: Multiple Hierarchies, Simple Markets, Multiple Markets, …



Organisation Dynamics: Open MAS

Dynamics of Organisations

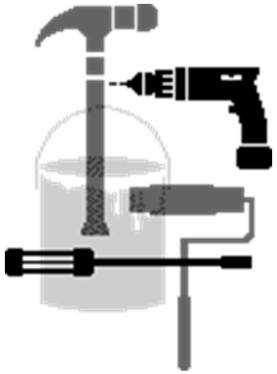
- The operational dynamic, which deals with how the states of the system elements vary in time
- The structural dynamic, which deals with how the states of the system structure vary in time

Open MAS: Multi-Agent Systems where agents can enter and leave freely. The structural dynamics is the central one here:

- Agent migrate.
- Agents may inaugurate new roles or new links in the system.

Developing MAS: Tools

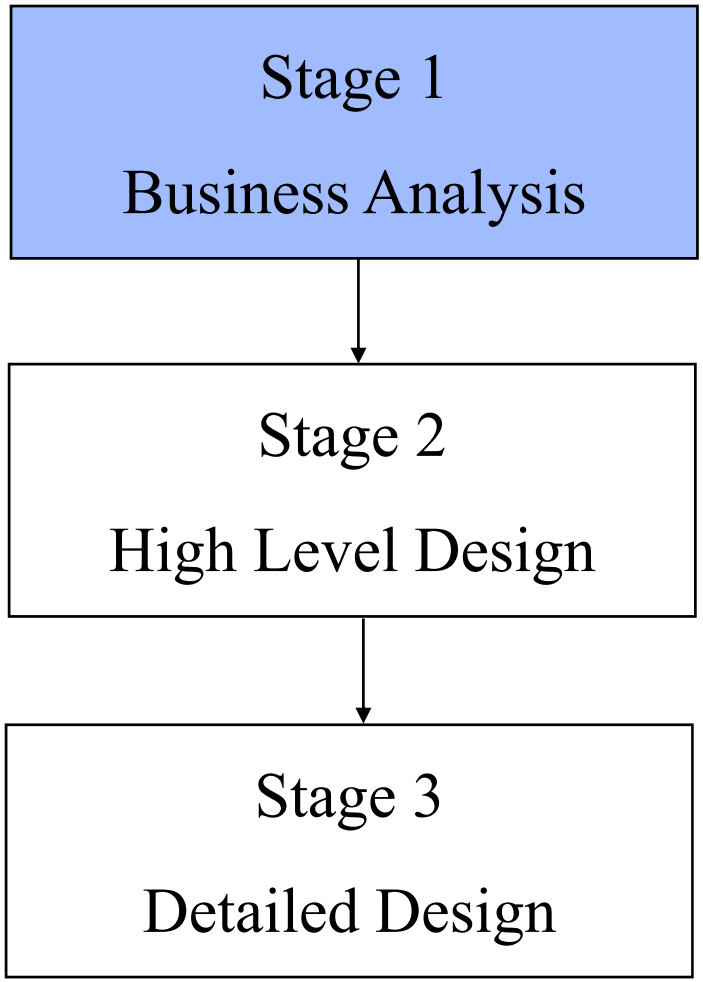
- MAS Design relies on existing languages and programming paradigms
- The trend of the work is towards MA-Oriented Programming, meaning programming MAS with MAS tools
- Example IDEs: AgentBuilder, AgentFactory, AGENTTOOL, 2APL Platform, Cougaar, Jack, JADE, Jason, JIAC, LS/TS, SPARK, JDE, VisualSoar, IMPACT, Zeus, etc.



Developing MAS: Methodology

- Needs a methodology that can cater to the notion of an "agent" (e.g. AAII, Gaia, Tropos, Prometheus, AUML, MESSAGE, MAS-CommonKADS, etc)
- The proposed Agent-Oriented Software Engineering (AOSE) methodology:
  - » Is based on the features combined from the existing methodologies
  - » Uses concepts from the UML notations loosely
  - » Suited for coarse grained agents, for situations where:
    - ♦ A group of loosely coupled systems (agents) are involved
    - ♦ The inter-agent relationships do not change at run time
    - ♦ Capabilities of the agents are static and do not change at run time
    - ♦ The interaction between the various agents is controlled by a business process

The AOSE Stages



- Determine how the business operates or would like to operate
- At the end of this phase, the deliverable is a complete domain description that describes:
  - » Who, How and What
  - » Which business process(es) need to be automated
  - » What are the steps of the business process(es) ?

Stage 2: High Level Design

Step 1	Macro Process Modelling	Macro Process Model » Use Cases that identify the actors and their interaction » The overall process as activity diagram
Step 2	Role Modelling	Role Model » Details of each role
Step 3	Elaborating Goals	Goal Elaboration Model » Details of each goal
Step 4	Domain Modelling	Coordination Model » Interaction between the various agents Ontology Model » Defines the objects that need to be exchanged
Step 5	High Level Architecture	Architecture Model » Schematic of how the system is composed, showing the individual agents and other components » Document describing the deliverables for all the above steps

Stage 3: Detailed Design

Step 1	Micro Process Modelling	Micro Process Model » The micro-process for each agent as activity diagram(s)
Step 2	Behaviour Modelling	Behaviour Model » Details of the behaviour(s) that must be implemented by the agent to satisfy the micro-process
Step 3	Platform Mapping (e.g. JADE)	Message Model » List of all messages and the participants involved in the exchange » Examples of messages Message Handling Model » Table showing the mapping between the message and the behaviour that will handle it Behaviour Instantiation Model » Diagram showing the inheritance of behaviours from behaviour classes provided by the platform Agent Class Model » Diagram showing the agent class diagram

Summary

- Distributed Intelligent Systems requires distributed problem solving techniques and a group of loosely coupled problem solvers (agents)
- The key is decentralisation (view, control, data, etc.)
- Agents communication: speak the same language!
- Co-ordination: a set of agents behave coherently => synchronisation, conflict avoidance/resolution, interaction protocols, MAS Organisations
- AOSE: methodologies for building Multi-Agent systems

The Holy Grail for MAS - Robotic Soccer

- Originated by Alan Mackworth (1993)
- As a general test-bed for MAS
- Gained popularity in recent years due to the RoboCup international competition
  - » Official Goal: *By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team.*
- Can be played by real robots or simulated ones







Reference

- Bond, Alan H., and Les Gasser, eds. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 2014.
- Wooldridge M. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2009.
- Sycara K. *MultiAgent Systems*. AI Magazine, 1998.
- White paper on Agent Builder (http://www.agentbuilder.com/Library/white_paper_r1.3.pdf)
- Finin T et al. *Specification of the KQML Agent-Communication Language*, Draft, Februrary, 1994.
- FIPA Agent Communication Language Specification: http://www.fipa.org/repository/aclspecs.php3
- Woodridge, Jennings N and Kinny. *The Gaia Methodology for Agent-Oriented Analysis and Design*, Autonomous Agents and Multi-Agent Systems, 3(3), 2000.
- N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien and B. Odgers, *Autonomous Agents for Business Process Management,* Int. Journal of Applied Artificial Intelligence 14 (2), 145-189, 2000.
- Iglesias, C., Garrijo, M., Gonzalez, J. *A survey of agent-oriented methodologies*. Agent Theories, Architectures and Languages, 1998.