



Sistemas Baseados em Agentes

Agent-based Systems

Responsible: António Silva

Email: ASS@isep.ipp.pt, apssilva@gmail.com

Web: www.dei.isep.ipp.pt/~asilva

Origin of Agent's concept

- In the 80's, the Artificial Intelligence community produced a new paradigm: the **Distributed Artificial Intelligence (DAI)**. It evolved from two areas: the **Artificial Intelligence** itself and the **Distributed Computing**.
- **Distributed Artificial Intelligence** definitions:

DAI's purpose is the problem solving in situations where a sole problem "solver", a single machine or computational entity doesn't seem adequate [Davis-1980].

DAI is related to the type of problem solving in which computation or inference are logically or physically distributed [Nilsson-1981].

- DAI evolved into two main areas:
 - Distributed Problem Solving
 - **Multi-Agent Systems**

Different views about Agents

- The use of the word “Agent” reaches its peak after the 90’s, with the **Internet** boom. A new class of applications using the agent’s concept made its appearance.
- The word Agent comes into widespread use, even if, at times, just as a **marketing** gimmick. A perfectly conventional program could therefore be presented as an Agent.

- The term Agent has been adopted by different scientific communities, besides Artificial Intelligence:
 - Distributed Systems
 - Object-Oriented Programming (Distributed Objects)
 - Robotics
 - Graphic Computation, Virtual Reality, Man-Machine Interface
- These communities stress Agents' characteristics which are different from the ones that are most valued by the A.I. field.

What's an Agent ? (1)

A dictionary search for the term Agent will reveal one of these meanings:

- 1) a person or thing that takes an **active role** or produces a specified effect;
- 2) a person or entity that **acts on behalf** of another;
- 3) a **means** used by some intelligent entity to obtain a certain result;

Weak notion of Agency

Agents are hardware or software based systems with the following properties [[Wooldridge](#)]:

- **Autonomy** – ability to operate without direct human intervention and to control its own actions
- **Social capacity** – ability to interact with other agents via an Agent Communication Language (ACL)
- **Reaction** – means to perceive the environment, physical or otherwise, where they exist and react timely to changes.
- **Proaction** – ability to show goal-driven behaviours, to take initiative.

Strong notion of Agency

views an agent as a computer system that, in addition to having the properties already identified, is designed and created using concepts that are usually applied to humans.

It is common in AI to characterise an agent using mental categories, such as knowledge, belief, intention, and obligation [[Shoham](#)].

Some AI researchers go even further, and conceived agents as capable of displaying emotions [[Bates](#)].

Objects versus Agents

- Degree of autonomy
- Flexible autonomous behaviour (reactive, pro-active, social)
- Single / multiple thread of control

**Objects do it for free,
agents do it for the money.**

[Jennings et al, 1998]

What's an Agent ? (4)

Other definitions:

Intelligent Agents are software entities that perform a set of **operations on behalf of a user** or a program, with a high degree of **independence** or **autonomy**, using some **knowledge** about the user's wishes and purposes. (IBM)

Agents exist in **dynamic** and **complex environments**, where they **feel** and **act autonomously**, and perform a series of **tasks** for which they were specially designed. [Maes-1995]

I will call **Mind Society** to a scheme under which each mind is made of a lot of little processes, called Agents, **each agent** being **only capable of performing simple tasks** that don't require any thought. Nevertheless, when we **put together these agents in societies** under special modes, this will **lead to real intelligence**. [Minsky-1986]

Emergent behaviours



V-formation rules

- Rule 1 (**coalescing** rule): Seek the proximity of the nearest bird.
- Rule 2 (**gap-seeking** rule): If Rule 1 is no longer applicable, seek the nearest position that affords an unobstructed longitudinal view.
- Rule 3 (**stationing** rule): Apply Rule 2 while the view that is sought is not obtained or the effort to keep up with the group decreases due to increased upwash.

- **Flocking** is a collective movement of multiple autonomous entities
- Collective behaviour shown by large groups of certain species of birds, fishes, insects and even mammals.
- Example of **emergent** behaviour that spontaneously results from the aggregation of multiple similar individuals that follow certain common simple rules, without any global coordination.

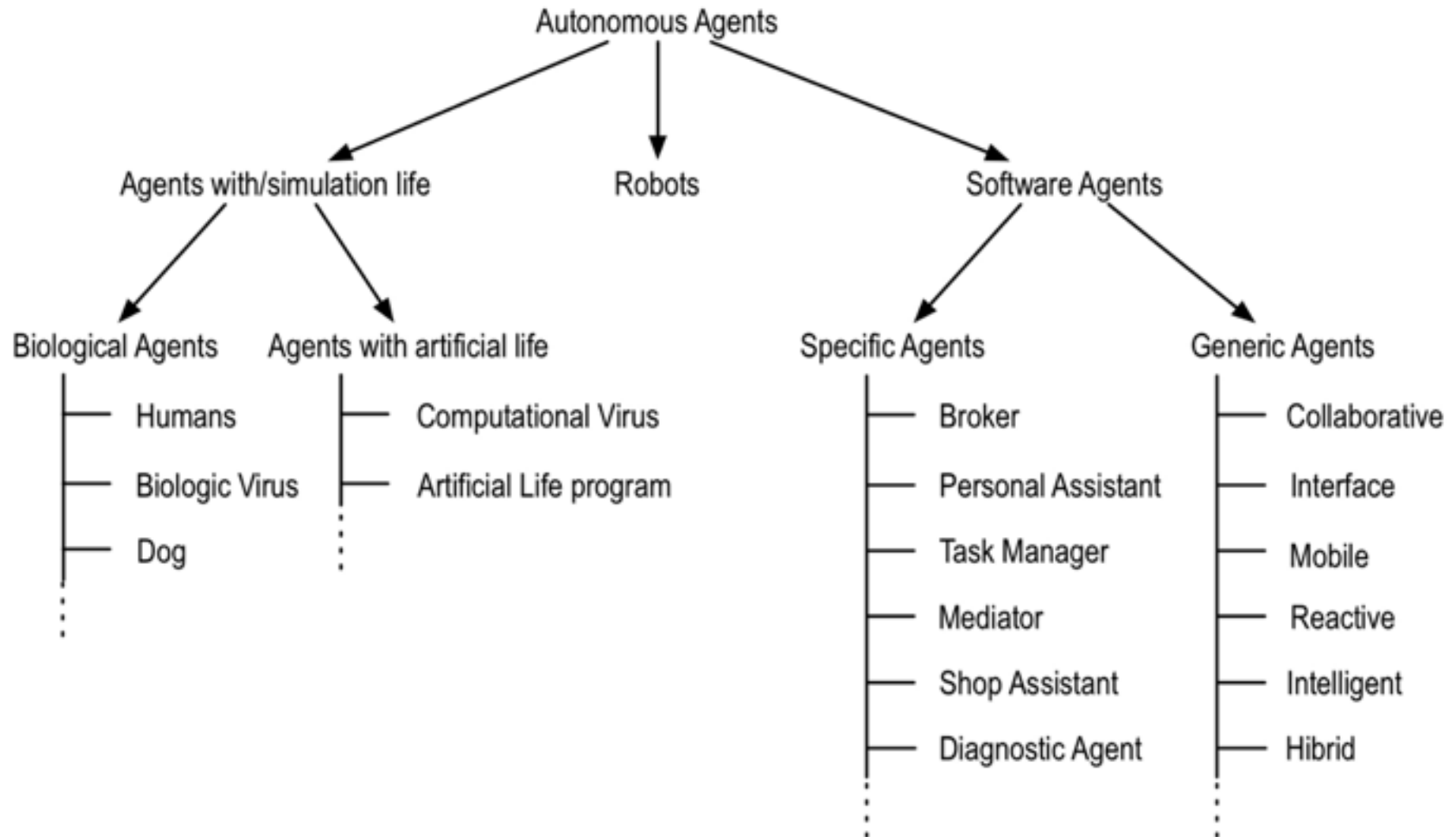
Basic models of flocking behavior are controlled by simple sets of rules, like:

1. **Separation** - avoid crowding neighbors (short range repulsion)
2. **Alignment** - steer towards average heading of neighbors
3. **Cohesion** - steer towards average position of neighbors (long range attraction)

Main Agents' Characteristics

Caracteristics	Description
Sensorial Ability	Have sensing mechanisms to get data about the environment
Reactivity	Feel and act, reacts to environmental changes
Autonomy	Decide and control their own actions
Pro-activity	Are driven by goals, don't act only in reaction to environmental changes
Persistency	Exist during a long period of time
Social skills	Are able to communicate and cooperate with other agents and eventually with people, to compete, to argue and to negotiate
Learning	Change their behaviour according to prior experience
Mobility	Are capable to transfer themselves from one computing environment to another
Personality	Show a credible personality and emotional behaviour
Intelligence	Are able to reason autonomously, to plan their actions, to correct errors and react to unexpected situations, to adapt and learn, to manage conflicts

Autonomous Agents Taxonomy (*)



(*) [Franklin-1996]

Autonomous agents

- For an Agent to act on behalf of someone, it must have a great degree of autonomy.
- Autonomy is universally accepted as the most important characteristic of an Agent.
- A fully autonomous Agent is an agent that doesn't need others to assure its existence or persistency.
- It will not freeze just because others (agents or human beings) were not capable of fulfilling a certain task.

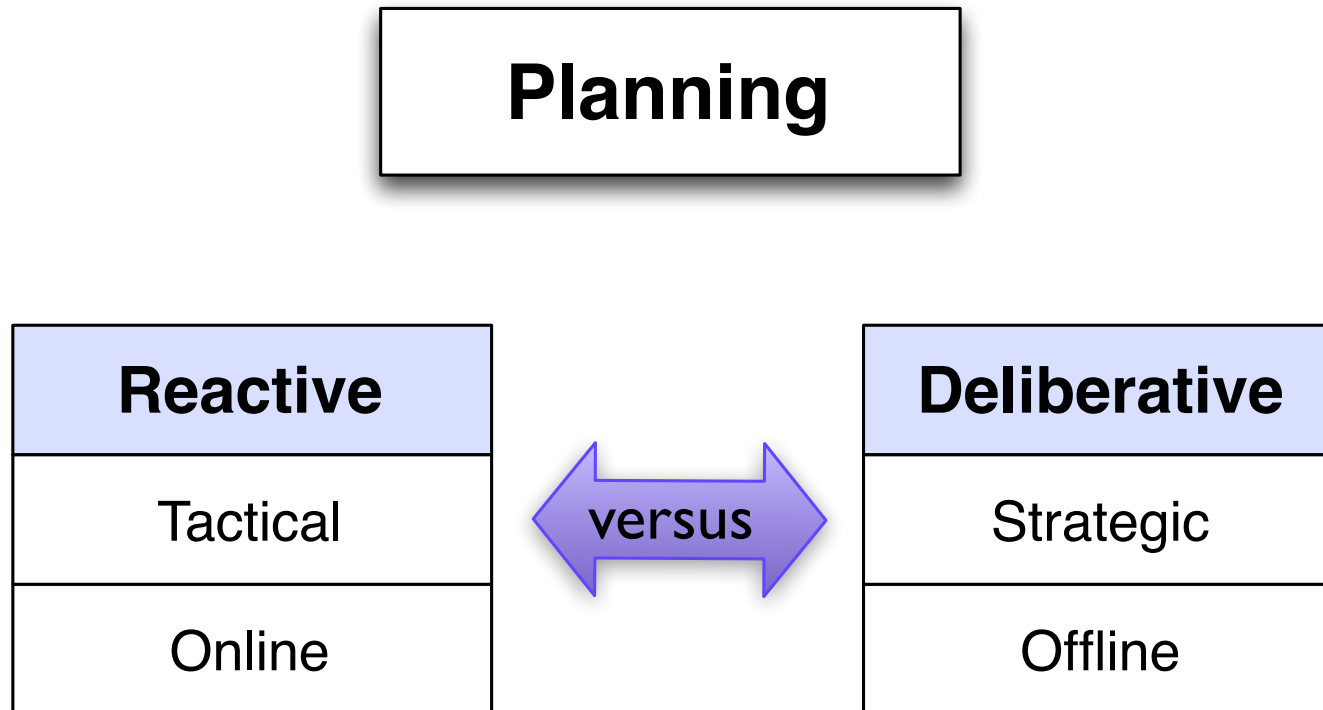
- “Intelligent” robots followed the cycle “**Feel**→**Plan**→**Act**”, where all actions were planned and response times could be high.
- Brooks' concept of **reactive behaviour** – the system should react to stimuli without a deep planning.
- Its architecture is based on the assumption that **Feel/Plan/Act** is not a cycle but made of tasks to be performed in parallel.

- In **highly dynamic environments** it is advisable to employ agents with reactive behaviours capable of reacting quickly to change. However...
- A complete inhibition of a deeper reasoning in favour of immediate reactions can also lead to problems.

Types of Reactive Systems

- **Purely Reactive Systems** – no planning, only reactive behaviours.
- **Reactive Systems** monitored and **controlled by planning** – in case of conflict, the planning module can take control over the actuators bypassing the reaction module.
- **Modifiable Reactive Systems** – the planning module can change or add reactive behaviours. May exhibit adaptive and learning capabilities.

Reactive & deliberative agents



- **Deliberative Agents** keep an internal representation of its environment, using an explicit mental state modifiable by symbolic reasoning.
- A purely deliberative hypothetical agent wouldn't change an initial plan just because the environment had changed.
- In practice, full reactive or deliberative agents are hard to find, most of them being **hybrids** closer to one or other end of the spectrum.

- In an **industrial production system** an agent representing a **production order** is [REDACTED], but an agent representing a production line is [REDACTED]
- An agent responsible for **alarm processing** in a power system Control Centre is a [REDACTED]. An agent in charge of planning the power system restoration after a serious incident can be [REDACTED].
- An **Electronic Commerce** agent representing a certain vendor will be a [REDACTED]. An Electronic Commerce agent representing an occasional buyer must be considered as [REDACTED]

- **Sociability** is key to differentiate between an intelligent software system and a system of intelligent agents.
- Agents must use a **common language** and share vocabularies and taxonomies in order to be able to understand themselves.
- A **cooperative** agent needs to know what its skills are and to have an idea about what tasks can be accomplished by other agents. These agents are able to share both tasks and results (data and knowledge).

Communities of cooperative agents can be divided in

- **Tightly coupled** systems - agents are very dependent on each other. If one agent fails there is a strong possibility that the multi-agent system also fails.
- **Loosely coupled** systems - agents have a greater autonomy. If an agent fails the system will be able to find a solution, although of lesser quality.

- Agents can also **compete** between themselves. In that case, agents must have increased abilities to watch closely its competitors.
- Agents with social skills should be able to **negotiate**. Negotiation is based on **announcements**, **proposals**, **offers** and **decisions** and is usually bound by **restrictions**, **conditions** and **penalties**.

Types of conflicts between agents

- Conflict of **Interest** – agents have different goals, eventually contradictory;
- Conflicts of **Responsibility** – different agents want to take responsibility for the same task;
- Conflicts of **Information** or **Knowledge** – agents have different views on the same situation or reality.

- **Agent Mobility** is defined as the ability to transfer itself to a different computational location.
- Not to be confused with the concept of software **portability** or **physical** mobility.
- A mobile agent can migrate from one machine to another in a **heterogeneous** environment.
- The agent chooses when and where to migrate to.
- It can suspend execution in a certain machine and transfer itself to another one, reactivating itself upon arrival and resuming its work.

- When agents move across a network they use **resources**. Attention should be paid to avoid the overuse or waste of these resources.
- Mobile agents must be able to deal with **heterogeneity**: an agent may visit machines of different types and operating systems, used by organizations with different policies and goals.
- Mobile agents platforms face issues of **fault tolerance, access priority** and **security**. Mobile agents can be vulnerable to hostile attacks.

- Work has been done in the development of **anthropomorphic computational personalities** that seem to exhibit some kind of emotional behaviour.
- A specific character, elements of personality or an emotive behaviour can be assigned to an agent.
- Work has also been made towards the **user's emotional status identification**.
- Agents with the ability to exhibit and recognize emotional behaviour need to rely on **voice recognition, natural language, computer vision** and graphical computation.

Questions (beyond the mere technological ones):

- What will be the real impact of an interface agent looking like a human face and seeming to exhibit some kind of personality or even feelings?
- Will the introduction of such an agent increase user's comfort and satisfaction with the interaction?
- Will such an agent be more persuasive?

Electronic Commerce

- Search Automation
- Safety beyond transactions (trust level)
- Individual Purchases
 - Agent Personalization
 - Standard or Differentiated Products
 - Seller Agents
 - Product advertisement and transaction security
 - Transaction analysis and competition monitoring
 - Customer profiling - Data mining

Electronic Commerce (cont.)

- Business2Business
 - Well defined products
 - Careful selection of suppliers
- Intermediary agents

Robotics

- **Picking and Assembly**

- SMA assuring functions like component identifier, trajectory planner, assembly planner and execution controller
- Essentially cooperative, heterogeneous agents

- **Surveying**

- Robot community
- Homogeneous, reactive, essentially competitive agents

Manufacturing Systems

- Highly complex, naturally distributed (logical & physical) environments
- Multi-Agent Systems
- Resources and Task Agents
- Cooperation and Negotiation at several levels
- Decentralized approach

Traffic Control

- Critical and complex distributed application
- Tasks and Resources agents
- Aerial or car traffic

1 - There is no system controller

- Agent-based solutions may not be adequate to problems where **global restrictions** must be held,...

...where a **real-time** behaviour is to be expected and...

deadlocks to be avoided.

2 - There is no global perspective

- An agent's action is determined by its **local** status, because the complete global knowledge is not attainable.
- Therefore, a agent-based system can only achieve **sub-optimal decisions**.

3 - Trust and Delegation issues

- **How can we be assured that the agent is correctly representing us?**
- One needs to trust an agent in order to delegate tasks on it.
- The building of trust on agents requires **time** and **experience**.
- Careful agent **personification** and **scalable intelligence** can ease this process.

Agent-based Systems (ABS) are a simple to **understand** but hard to **implement** concept. Do not overestimate its potential.

Agents as a **dogma**, to see agents everywhere, when OOP may be adequate

Agents should be designed for **specific** applications, we shouldn't try to make them too generic.

ABS development requires **Software Engineering** methods, as any software does.

Confusion between **prototypes** and **systems**

– real SMA deal with complex aspects:

- Distributed and concurrent problem solving;
- Flexible and sophisticated interface between components;
- Complex individual components with context-dependent behavior.

Forgetting that ABS are **Distributed Systems** only even more complex...

Concurrency, one of the main advantages of DS like the SMA is not used

“**Not Invented Here!**” syndrome, trying to develop infrastructures and platforms from scratch

Avoid **extremes**:

- **Degree of intelligence**

- Too much AI may impact system robustness
- Too less AI may turn agents into dumb objects

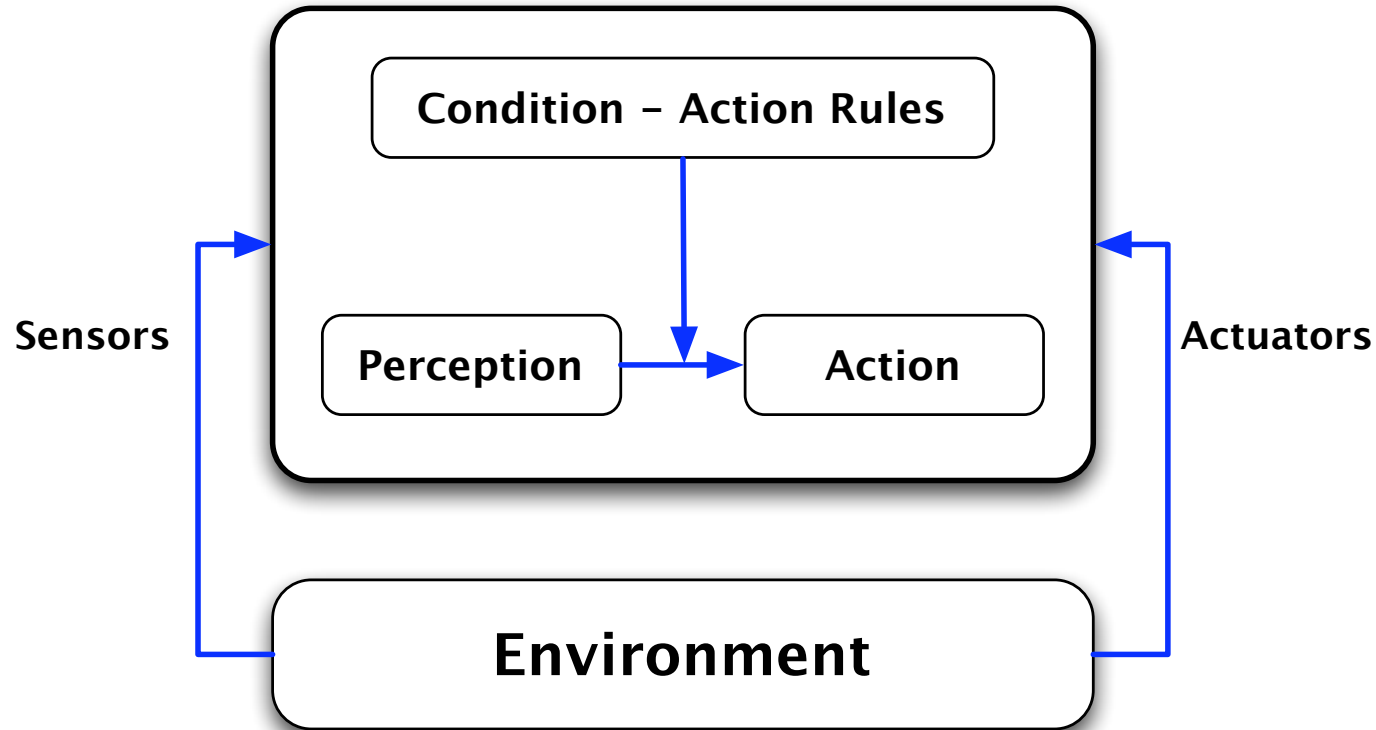
- **Number of Agents**

- Too many agents may lead to emergent behaviors or ... chaotic situations
- Too few agents limit concurrency

Insufficient **Normalization** (KIF, KQML, ACL start to emerge)

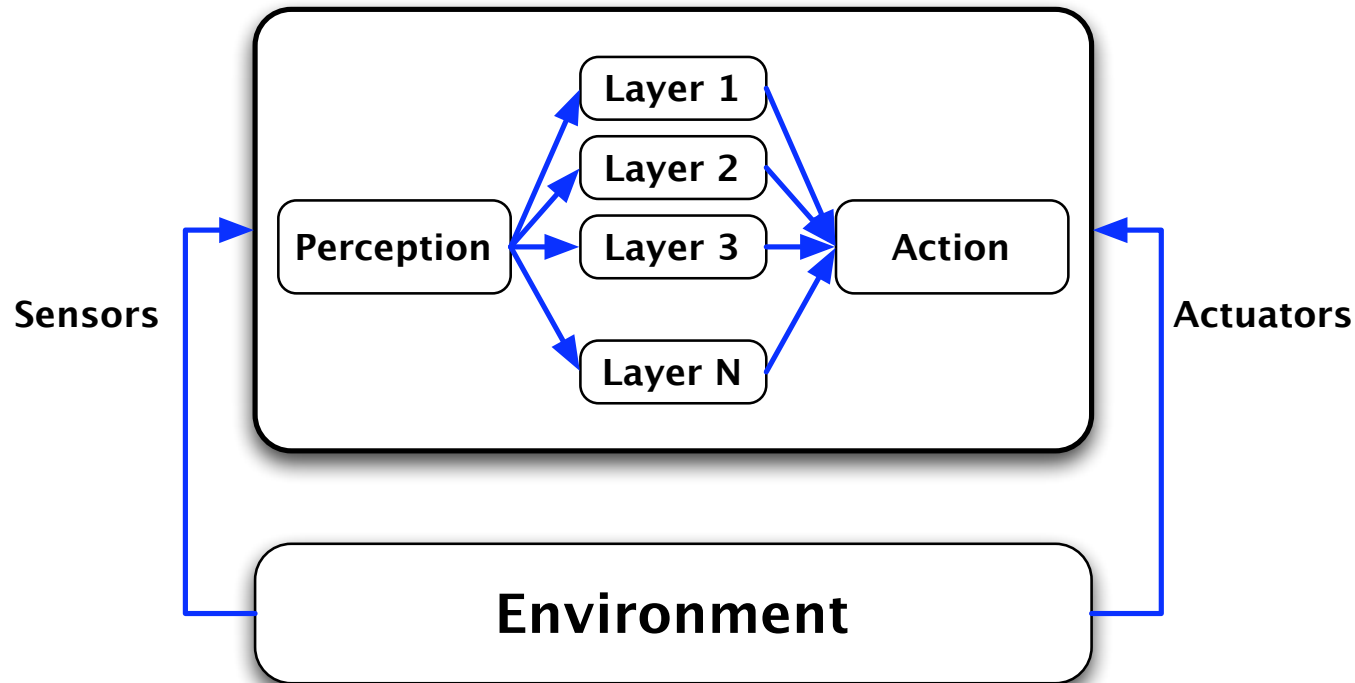
An agent's **architecture**

- determines its internal **structure**,
- defining the **modules** involved in the various tasks and
- the **interaction** between those modules



Task-driven behaviours

Reactive architectures



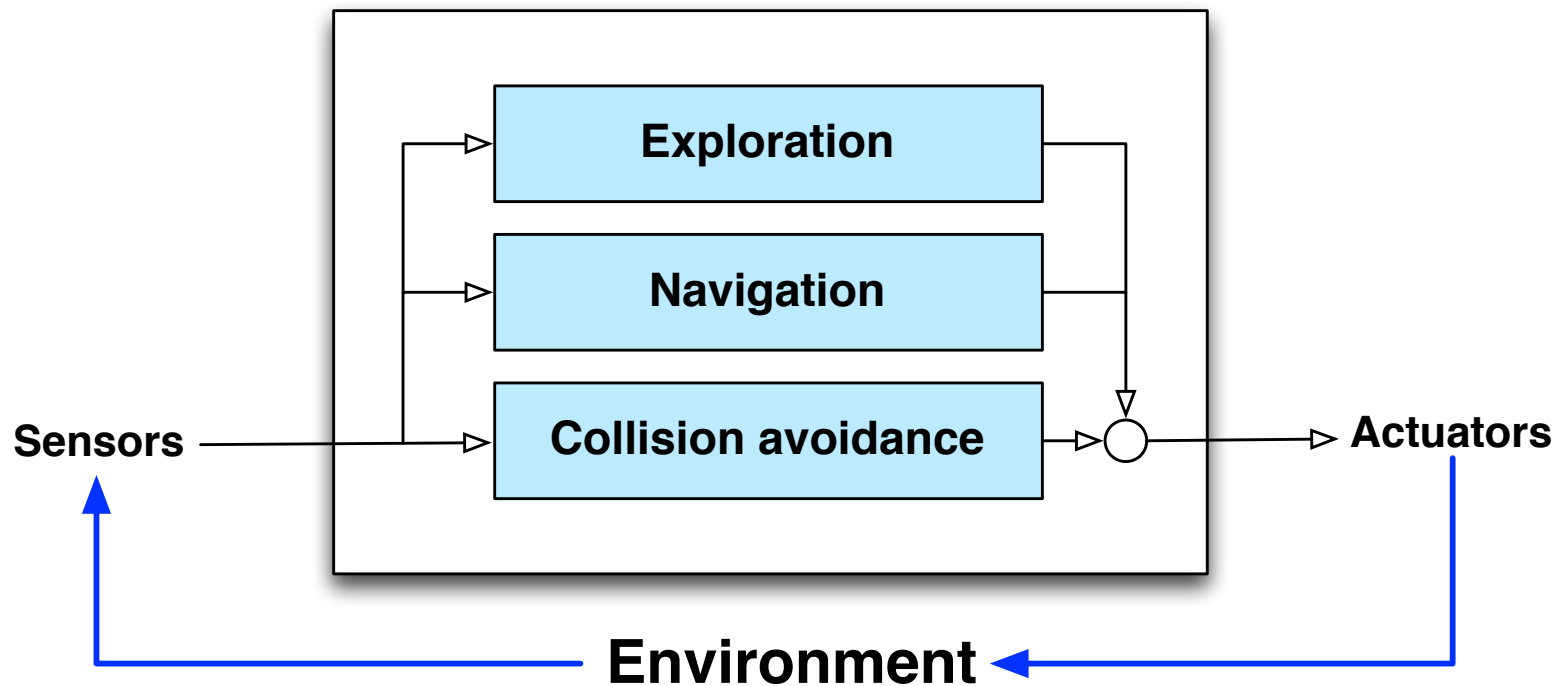
- + **Simple and Robust**
Computationally treatable
- **Incapable of learning**
Limited nr of behaviours

- A **subsumption** architecture [*] is based on the decomposition of complex intelligent behaviours into many basic **behavioural elements**, organized into **layers**.
- Each layer implements a specific goal of the agent, and layers are organized in increasingly abstract order.
- Each layer's goal subsumes those of the lower level layers.

* Brooks, 1986

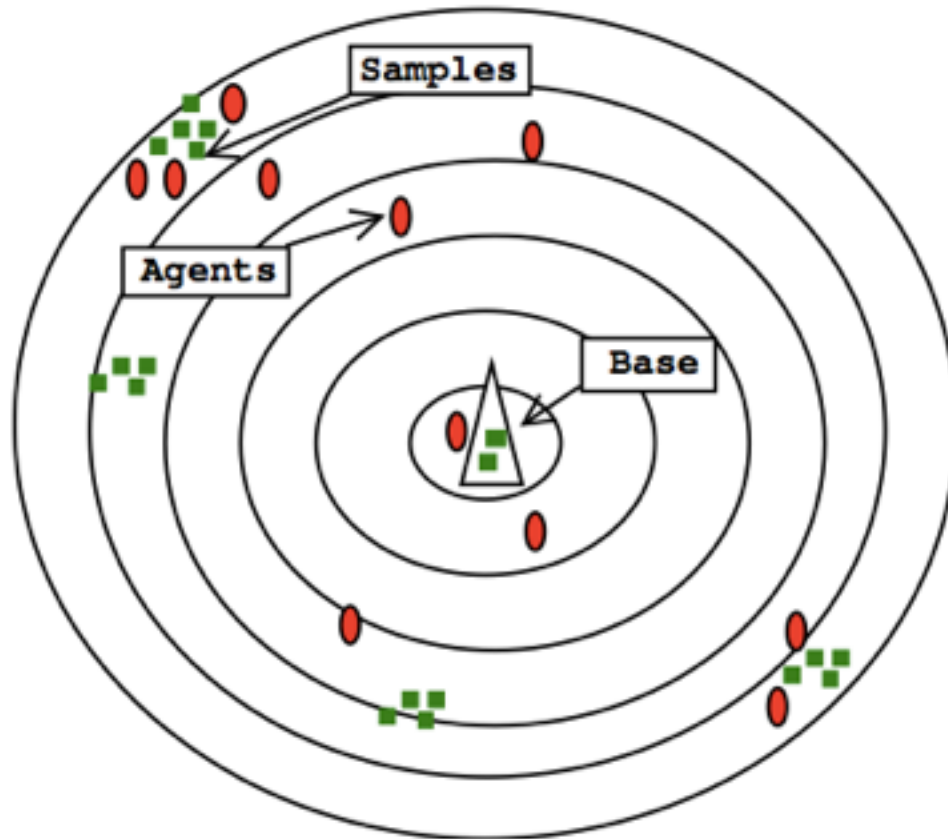
Example of subsumption architecture

Reactive agent controlling a robot



Example

- A robot's lowest layer could be tasked with **collision avoidance**, the next would be dedicated to **navigation**, all both under an upper **exploration** layer.
- Each of these layers receive the sensor data and control the actuators but **separate tasks can suppress inputs or inhibit outputs**.
- **The lowest layers** can then **work like fast-adapting reflexes**, while **the higher layers** are **dedicated to more global goals**.
- The feedback to agent initiated actions is given by the environment.



Collective problem solving

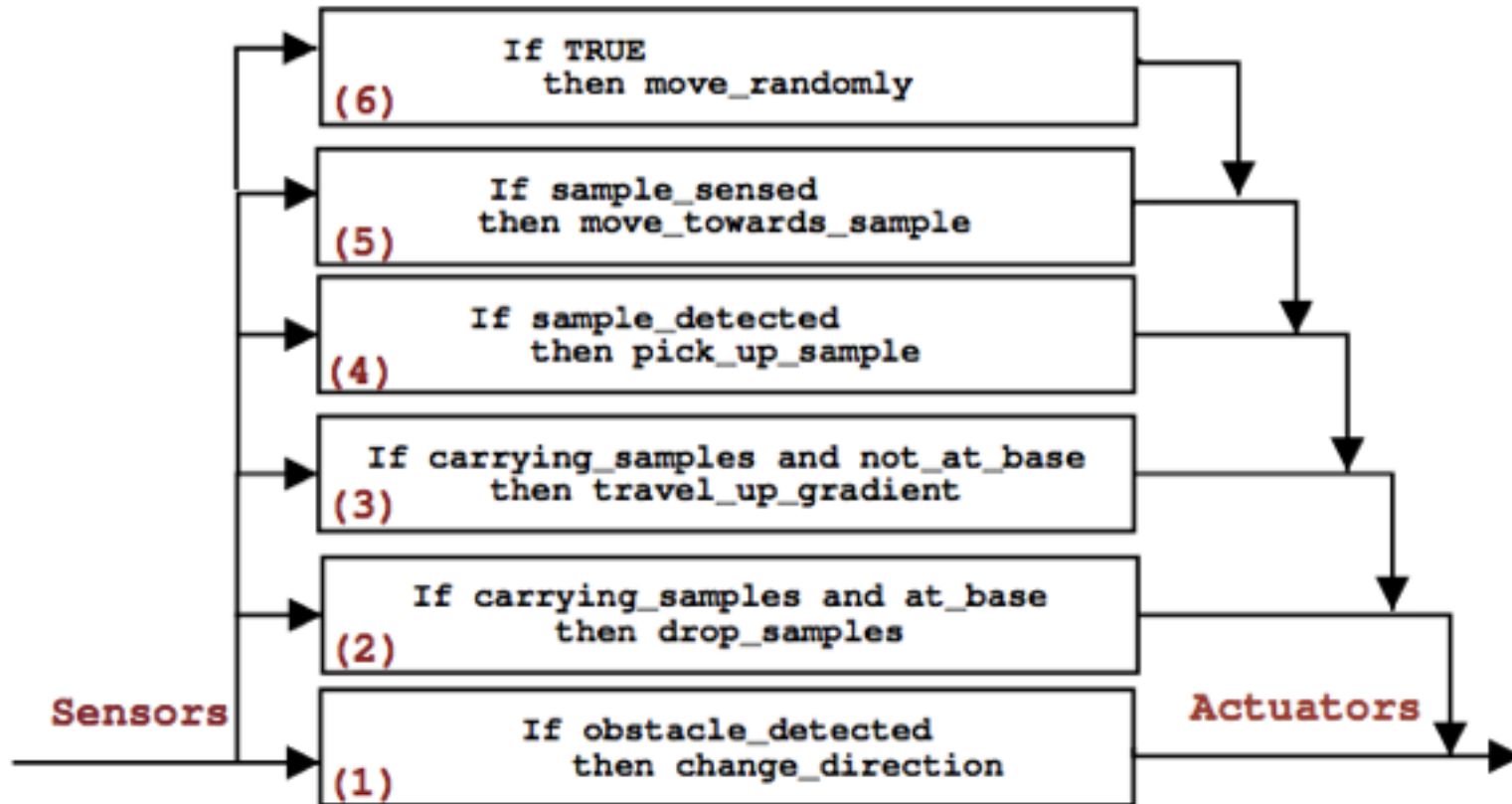
Case study: Foraging Robots

[Steels(1990), Drogoul and Ferber(1992)]

Constraints:

- No message exchange
- No agent maps
- Obstacles
- Gradient field
- Clustering of samples

Simple collecting behaviour



Subsumption ordering: (1) < (2) < (3) < (4) < (5) < (6)

Limitações

- Como não dispõe de modelo do seu ambiente, um agente reactivo deve possuir suficiente informação local;
- Como se baseia em informação puramente local (current state), está limitado a visão de curto prazo;
- Dificilmente poderá aprender com a experiência;
- Não há qualquer ligação entre eventuais comportamentos globais emergentes e os comportamentos individuais. Não existe metodologia, apenas tentativa e erro.

Deliberative Architectures (BDI)

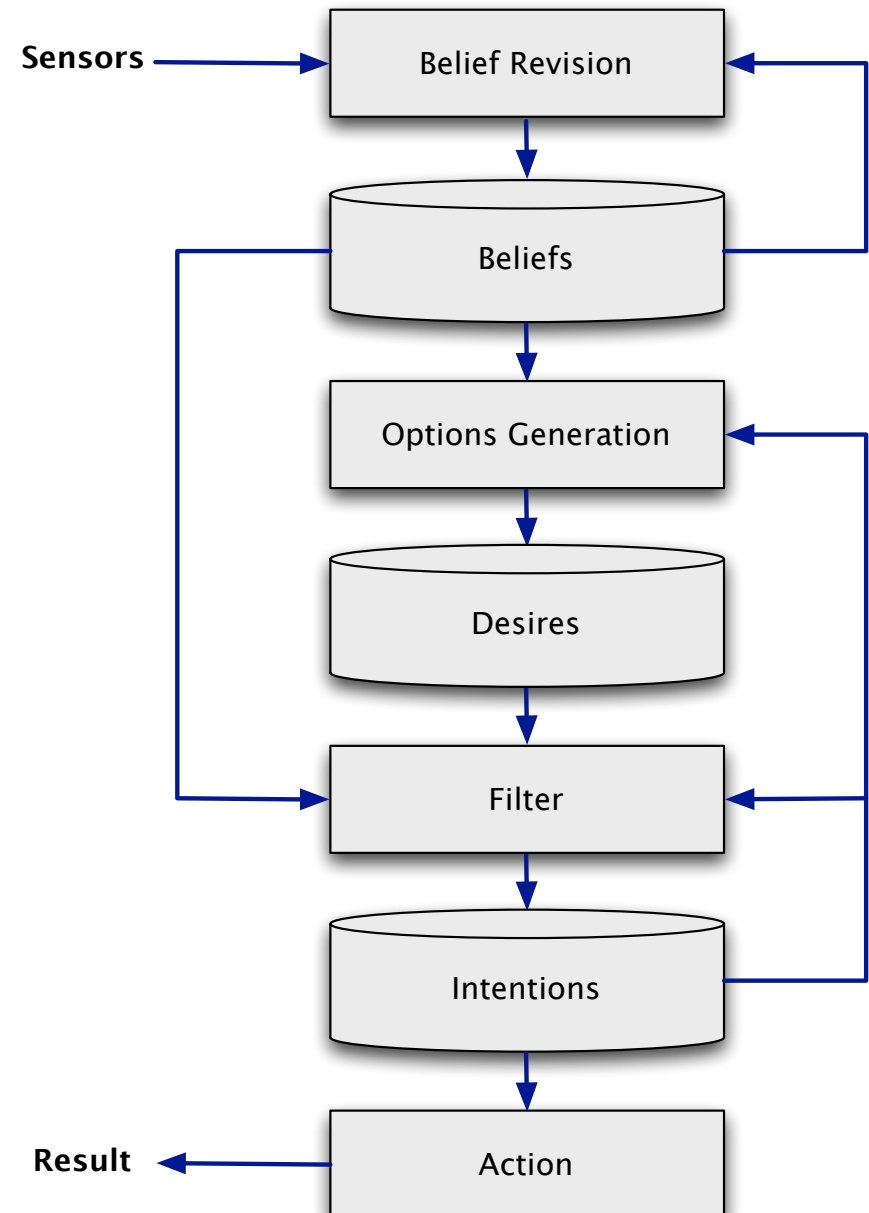
Beliefs – expectations that the agent has about its environment

Desires – preferences about future environments states

Goals – plausible desires

Intentions – goals that have been set by the agent and which will be the result of actions

Plan – pragmatic implementation of intentions



Intentions

- drive means–end reasoning
- constrain future deliberation
- persist
- are related to beliefs about the future

How to achieve a good balance?

By periodically reconsidering its intentions

But, reconsideration has costs...

An agent that

- doesn't often reconsider its intentions risks attempting to achieve intentions after they are no longer pertinent.
- constantly reconsiders its intentions may never achieve them

We need a trade-off!

It will be dictated by the environment

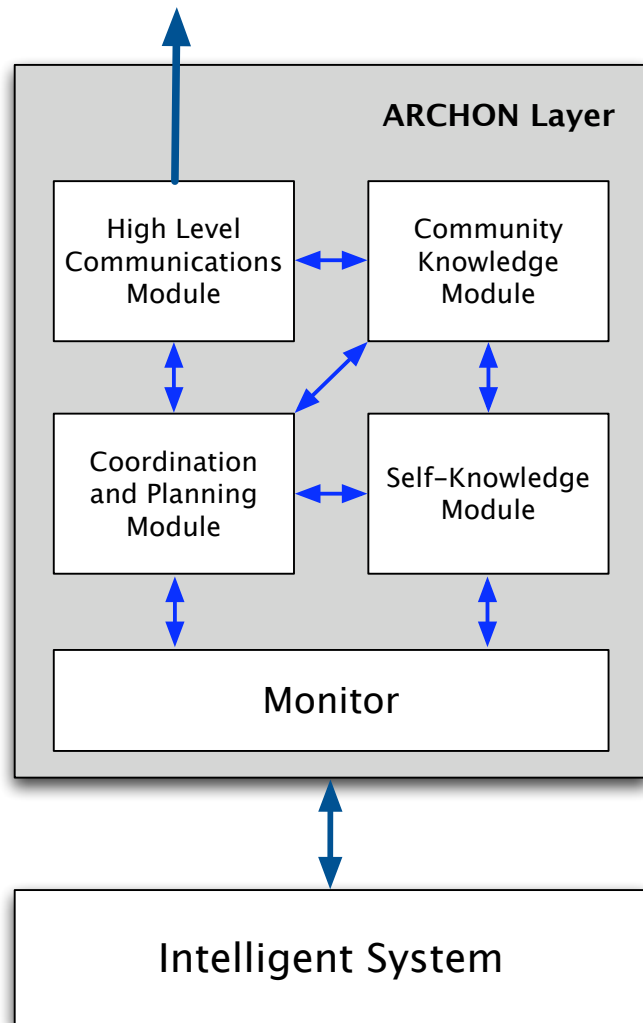
Rate of world change	Bold / Cautions Agents?
Low	Bold
High	Cautious

PRS – Procedural Reasoning System

Applications:

- Fault diagnostic for the Space Shuttle's reaction control system
- Air traffic management system at Sidney airport

ARCHON Architecture



Basic principle – any pre-existent system (Intelligent System) may be **encapsulated** by an ARCHON layer so that it may turn into an agent.

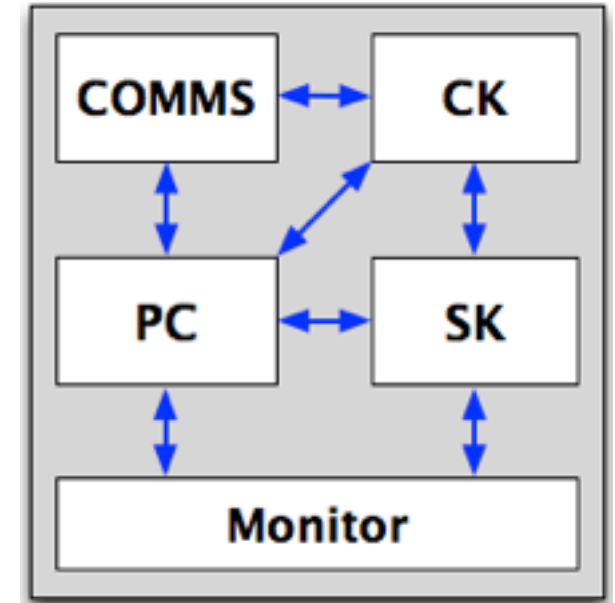
An agent will always have at least two components:

the **Intelligent System** – responsible for the useful work to be done by the agent;

the **ARCHON layer** – responsible for the cooperation with the other agents in the community and to control the Intelligent System.

Agents' Internal Architectures

- **Self-knowledge (SK)** – knowledge about itself and the tasks to perform;
- **Community knowledge (CK)** – knowledge about the other agents belonging to the community;
- **Planning and Coordination (PC)** – decides when and how the Agent establishes a cooperative relationship with the other community agents. Responsible for the **global assessment** and the dynamic **planning** of the agent's activities.
- **Monitor** – interaction with the Intelligent System (IS) and control of its activities. Receives requests and data from PC, schedules the IS tasks, receives its results and gives them back to PC;
- **High-level communication (COMMS)** – defines the dialogue between the Agent and the agent community. Uses a message passing mechanism plus intelligent addressing, filtering and message scheduling.



NOTE: Both SK and CK contain both static and dynamic knowledge

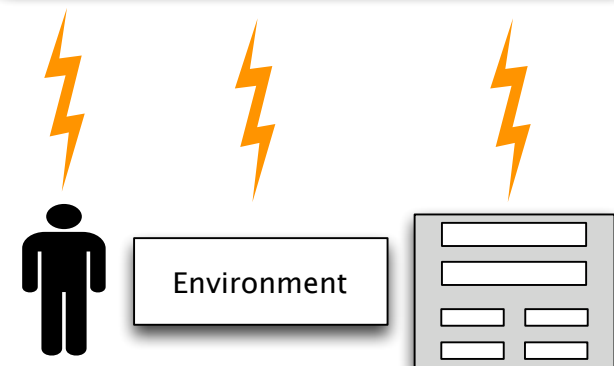
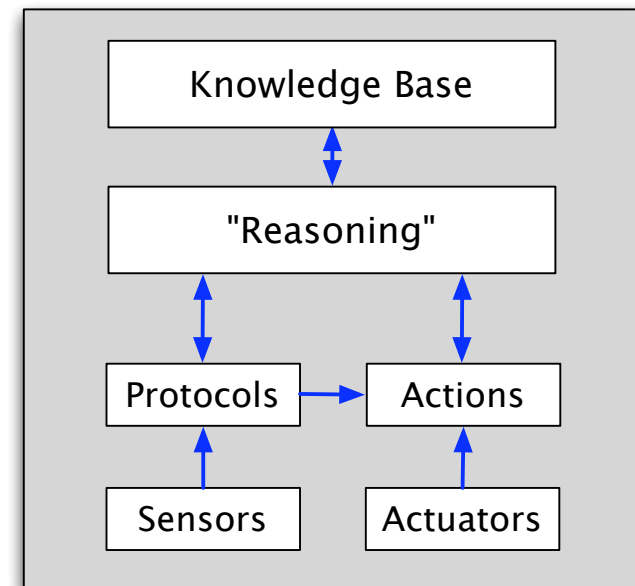
Agents' Internal Architectures

The term **holon** comes from the combination of the greek word **holos** (“whole”), with the english suffix **on** (“part”). It refers to the **whole and the part**, betraying a recursive nature: **a holon may be made of holons and be part of one** (or several ones).

The concept has been adopted by the **Intelligent Production Systems** community as a model suitable to describe a **Production System**.

The holon/agent is capable of **interacting** with humans, the environment and other holons.

Holonic Architecture



Agents' Internal Architectures

Sensors and **Actuators** that represent the system interface, enabling the interaction with humans, the environment and other holons,

Protocols that handle the representation of the information gathered by the sensors (**perception**). They may be identified by a finite state machine of a communications protocol or a man/machine interaction. It allows the **direct execution** of

Actions and the knowledge processing by the

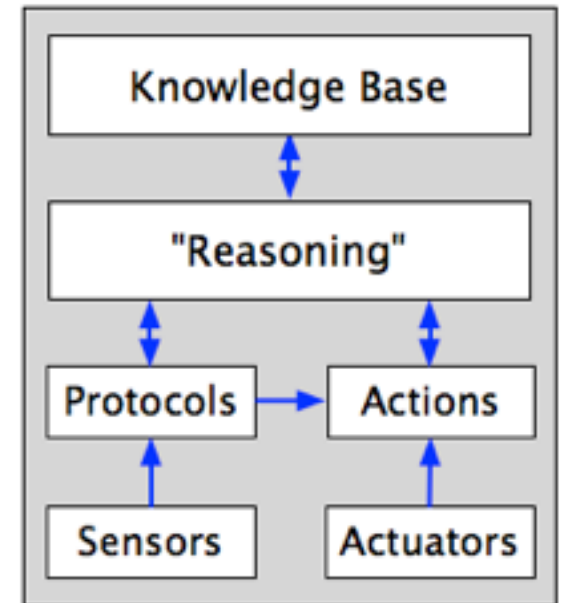
Reasoning block that produces results using data from the sensors and its

Knowledge Base. It defines the holon's nature, specifies how it should behave according to its mental state and its goals. The holon's knowledge may

be **inherent** to the holon's conception,

may be **learned** from experience or observation or

may come from **other holons**.

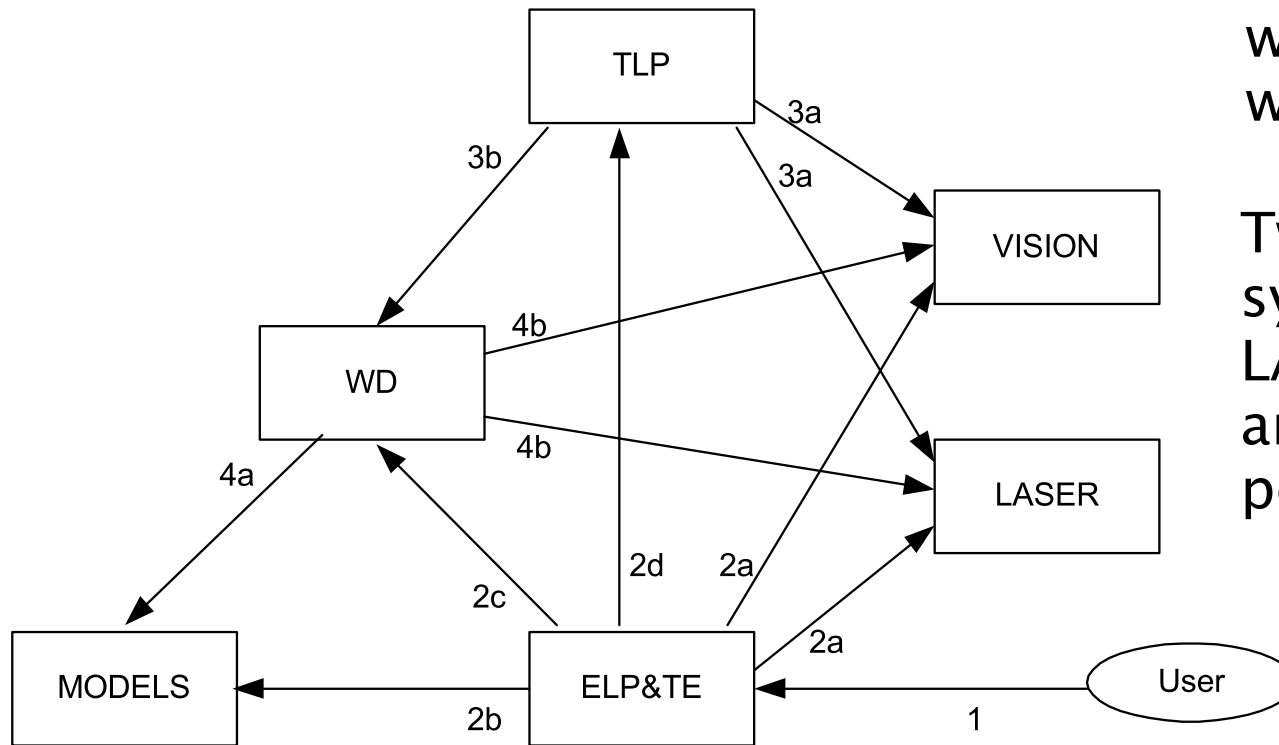


A Multi-Agent System architecture describes the relationships between agents looking for a solution for a given problem.

- Specific or Generic
- “*Centralized*” or Distributed
- Fixed or Reconfigurable
- Homogeneous (competitors) or heterogeneous (complementary) agents

Assembly Robotics

CIARC



Used in an assembly and manipulation system with a robotic handler with a articulated arm.

Two computerized vision systems VISION (2D) and LASER (3D) to recognize and identify objects' position and orientation.

TLP - Task Level Plan

WD - World Descriptor

ELP&TE - Execution Level Planner + Task Executor

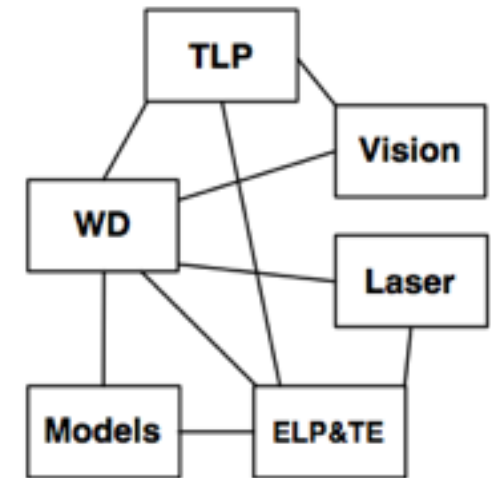
WD (World Descriptor) – capable of establishing **symbolic relationships** between objects (for ex., an object is on top of another);

TLP (Task Level Plan) – capable to generate **high-level symbolic plans** to manipulate objects (like inserting A into B);

ELP&TE (Execution Level Planner & Task Executor) – **Reactive** agent that controls the robot. It is capable of **geometric reasoning** in order to materialize the symbolic operations issued by the TLP agent.

MODELS – this agent stores several object models which are useful for creating symbolic relationships and the execution of assembly and manipulation tasks.

Excluding LASER and VISION agents, that have identical functions, all the others were functionally different.



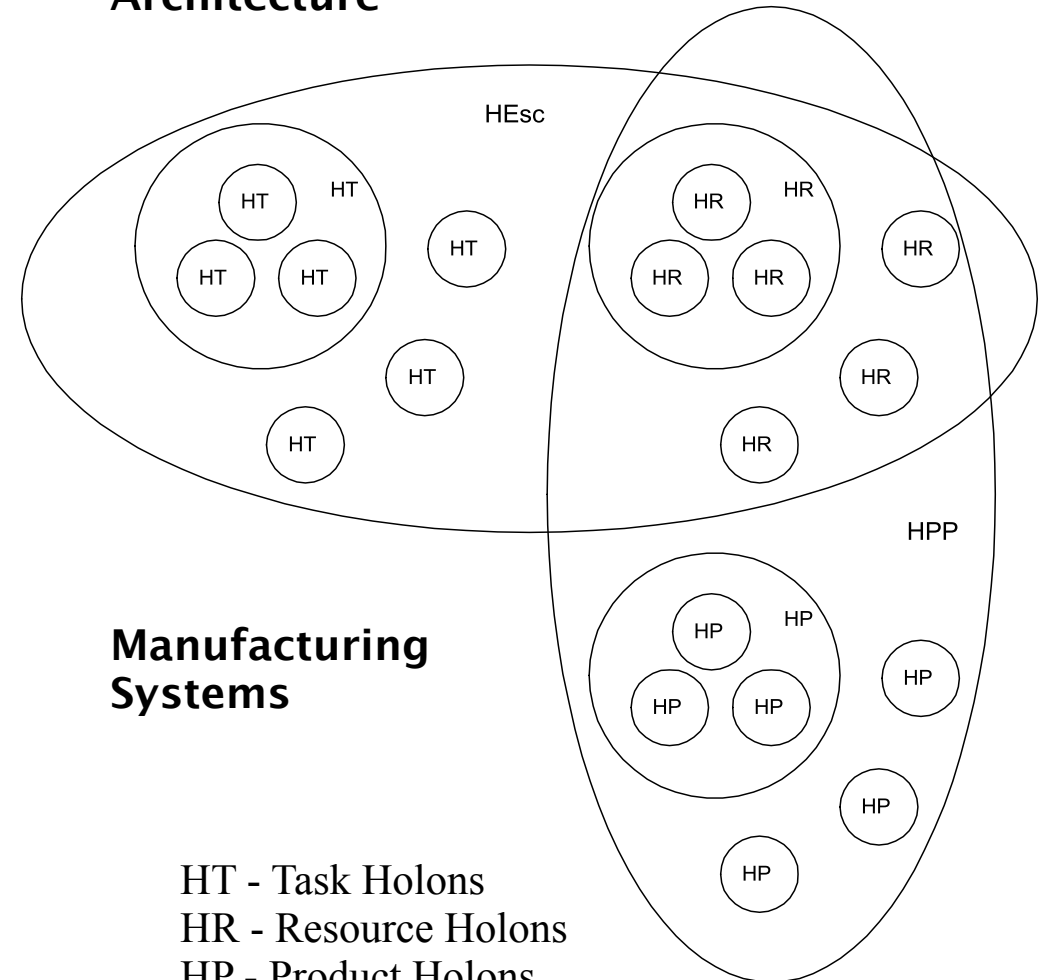
Holons representing **tasks** (HT), holons representing **resources** (HR) and others representing **products** (HP).

Basic **task** holons **get together** to form task holons of greater dimension or, a task holon can be **decomposed** in several more basic task holons.

Resource holons may be decomposed or be part of other holons.

Product holons can be made of component holons (can also be products).

Holonic Architecture



Manufacturing Systems

HT - Task Holons
 HR - Resource Holons
 HP - Product Holons
 HEsc - Holon de Escalonamento
 HPP - Process planning Holon

Phases of the production process involve different holons:

- **Scheduler** holon (**HEsc**) is composed of task holons and resource holons;
- **Process planning** holon (**HPP**) is composed of resource holons and product holons.

A holonic organization has a **hierarchical** structure (**holarchy**). The holarchy defines the **cooperation style**, subjecting the holons to pre-defined goals and limiting their autonomy.

1. Communications
2. Security
3. Directory
4. Conversation

Communication services

Message Exchange

Point to point

Bilateral message exchange

Agent knows with whom to communicate

Group message exchange

Multi-cast

Broadcast

Non-directed messaging

Communication services

Synchronism

Synchronous communication

Receiver stops and waits for message at a pre-defined moment

Efficiency problems

Agent may freeze if message never comes

Asynchronous communication

Communication and processing are independent

Received messages are stored in a queue

Robust and efficient

Communication services

Pooling

Storage of the messages and other relevant information waiting for the agent to recover.

Fault-tolerance

Forwarding

Receives the information the agent wants to send and takes care that it will arrive to the intended recipient

Security services

There is no central entity responsible for keeping the whole system's consistency.

System's security level is equal to the smaller security level of all the system components.

Names service

Before being granted access, an agent must pass verification phase.

Names service stores identification and passwords.

Permissions service

Membership doesn't give permission to do all

Divide agents in groups

Directory services

Information or Directory services inform an agent about other agents capabilities and contacts.

Information is usually **dynamic** – new agents register and provide their data. Centralized system is more robust.

Directory service as **Yellow Pages**.

Can assume two forms:

Facilitator – responsible for publishing and distributing information about each agent's services or tasks and contacts.

Broker (Discovery) – will search, when requested, for some particular information, using other information services or questioning agents about their capabilities.

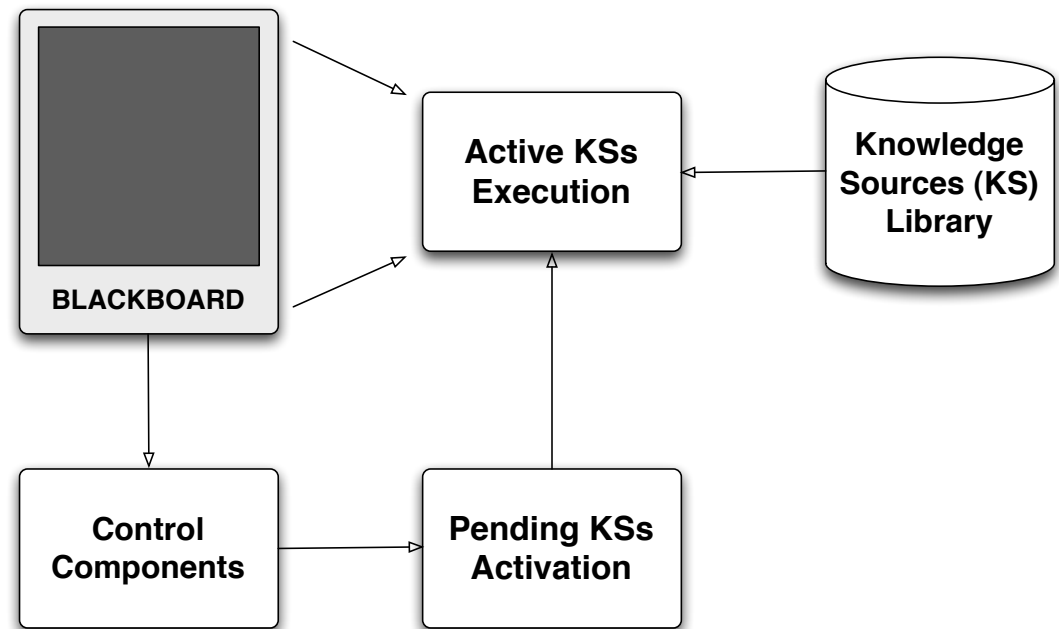
Communication services

Blackboards

Knowledge Sources (KS)

Shared Memory

Events - posting of requests
and answers



“A group of **specialists** are seated in a room with a large **blackboard** ... working as a team to brainstorm a solution to a problem, using the blackboard as the **workplace for cooperatively developing the solution**.

The session begins when the problem specifications are written onto the blackboard.

The specialists all watch the blackboard, looking for an opportunity to apply their expertise to the developing solution.

When someone writes something on the blackboard that allows another specialist to apply her expertise, she records her contribution on the blackboard, hopefully enabling other specialists to then apply their expertise.

This process of adding contributions to the blackboard continues until the problem has been solved.”

[Engelmore, 1988]

Characteristics of Blackboard systems

1. Independence of expertise
2. Diversity in problem-solving techniques
3. Flexible representation of blackboard information
4. Common interaction language
5. Positioning metrics
6. Event-based activation
7. Need for control
8. Incremental solution generation

Blackboard Systems
Daniel D. Corkill

Why use the Blackboard Problem-Solving Approach?

- Many diverse, specialised knowledge representations are needed.
- An integration framework is needed that allows for heterogeneous problem-solving representations and expertise.
- The development of an application involves numerous developers.

Blackboard Systems
Daniel D. Corkill

Conversations

A **conversation** is an ordered set, not necessarily sequential, of messages that are mutually understood by the intervening entities.

It systematizes occurrences and types of messages. At a certain moment, the set of possible messages is limited, mutually understood and correctly used.

Conversations

Timeout mechanism

Timeout should depend on the conversation state and participants.

Information management

Storage for the conversation state, the intervening agents and previous messages data.

Synchronization

Conversation control activities that manage the moment and order of message processing.

Negotiation is a process of Agent interaction with the purpose of reaching a mutually beneficial agreement.

It involves

information exchange,
mutual **concessions** and
relaxing of goals.

Main features

- ★ Language
- ★ Protocols
- ★ Decision process

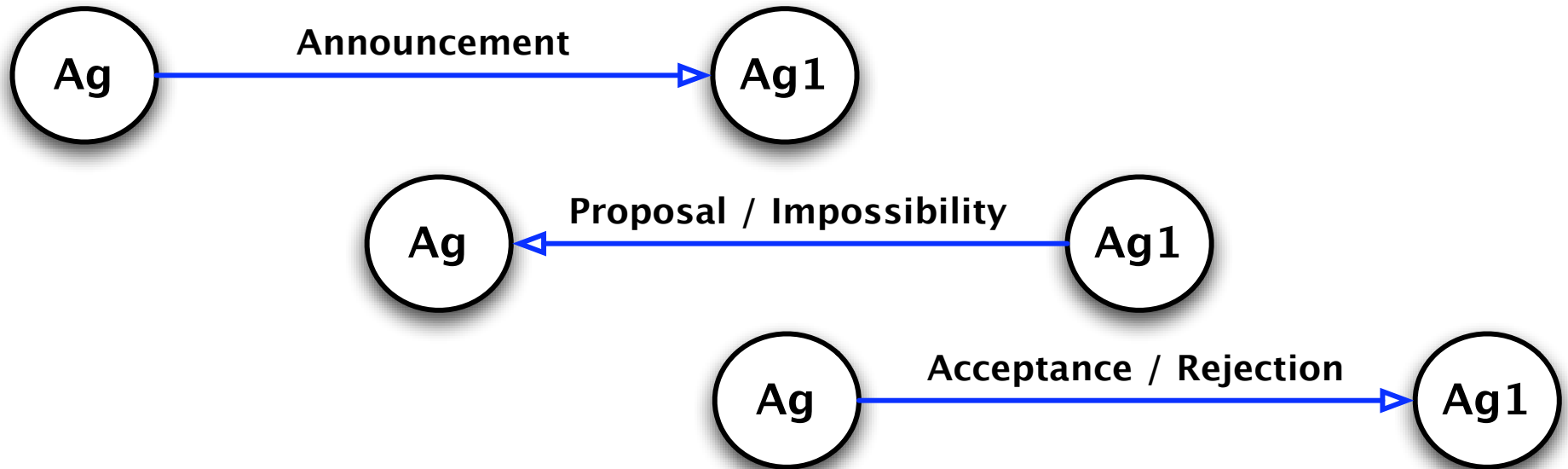
Negotiation Types

- ★ 1 contracting entity to 1 contractor
- ★ 1 contracting entity para N contractors
- ★ N contracting entities para 1 contractor
- ★ N contracting entities para M contractors

Atributos do mecanismo de negociação ideal:

- **Eficiência** no uso de recursos
- **Estabilidade** – um agente não deve ter incentivo para quebrar algo pré-acordado
- **Simplicidade** – baixas exigências computacionais e de largura de banda
- **Distribuição** – não requerer um árbitro
- **Simetria** – não estar enviesado em relação a nenhuma das partes

1 to 1 Negotiation



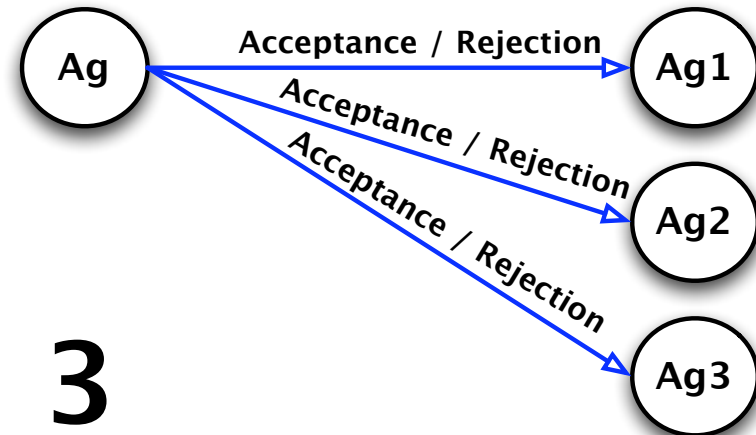
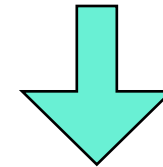
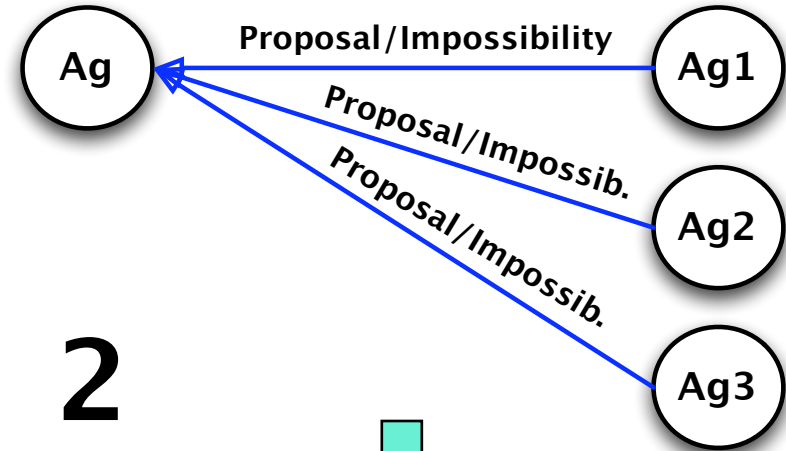
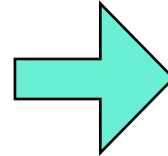
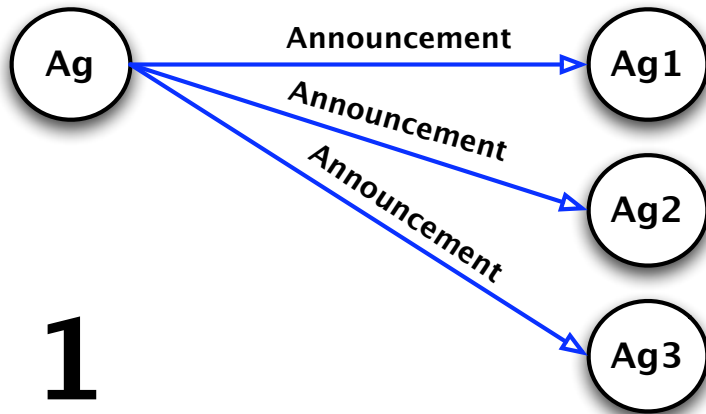
- Instance of Client-Server relationship (but contractor may refuse the tasks...)
- **Ag** should be able to handle impossibility
- Fault Tolerance mechanisms
- Process can be more complex and require extra interaction

1 to N Negotiation

Contract Net Protocol

- Interaction protocol for co-operative problem solving.
- Modelled on the contracting mechanism used by businesses to exchange goods or services.
- Offers a solution for the connection problem – find someone to perform a task for me.

1 to N Negotiation



Contract Net Protocol

- Meta-Knowledge about each Agent's capabilities needed
- Impossibility/Rejection messages may not be mandatory
- Fault-tolerance mechanisms

No CNP, um agente pode ser **contratado** e **contratante** ao mesmo tempo – um contratado para uma tarefa pode tornar-se contratante ao solicitar ajuda de outros agentes em partes dessa tarefa.

Estrutura dum anúncio:

- Agente endereçado
- Critério de elegibilidade
- Descrição da tarefa
- Especificação da proposta a submeter
- Expiração do anúncio

Negotiation Protocols

Agent **Ag** wants 3 **tasks** ($T1$, $T2$ and $T3$) to be performed with the following temporal restrictions:

$T1$ (duration - 2 time units) must precede $T2$

$T2$ (duration - 2 time units) must precede $T3$

$T3$ (duration - 3 time units) must be concluded before the instant 10 (10 time units)

Agents **capabilities** are the following:

$T1$ may be executed by $Ag1$ or $Ag3$

$T2$ may be executed by $Ag2$ or $Ag3$

$T3$ may be executed by $Ag3$ or $Ag4$

These agents' **agendas** are the following (in time unit intervals (t_{init}, t_{end})):

$Ag1$: [(1,2)]

$Ag2$: [(3,4)]

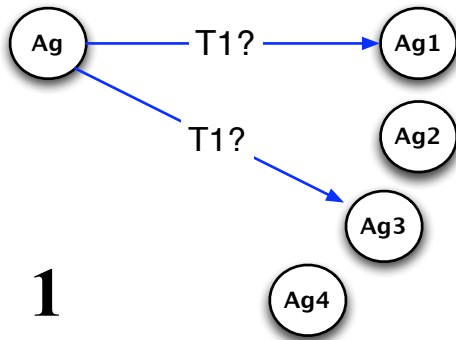
$Ag3$: [(1,3),(5,8)]

$Ag4$: [(4,5),(9,10)]

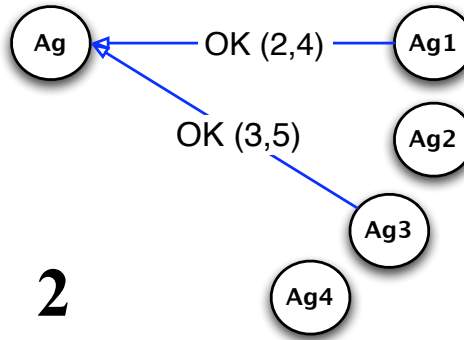
A possible negotiation **sequence** should be devised in order that Ag may be able to schedule the required tasks with the agents that are fit to perform them.

- ▶ This problem requires a **1 to 3 negotiation** for **each** task (T1, T2 and T3).
- ▶ These negotiations are **inter-dependent**.
 - ▶ The agent Ag3, for instance, may execute the T1 and T2 tasks. It may happen that this agent is awarded the execution of both tasks.
 - ▶ The negotiation of T2 may be influenced by the negotiation of T1.
- ▶ This may bring problems to the **simultaneous negotiation** of tasks.

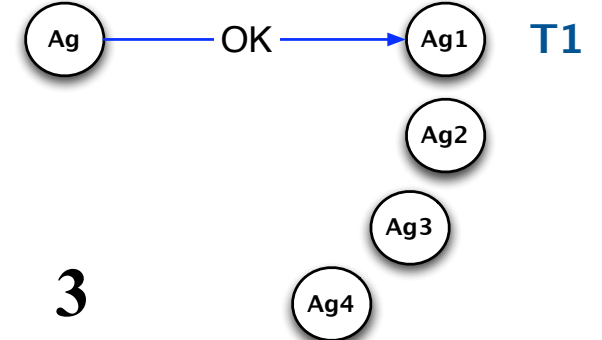
Negotiation Protocols



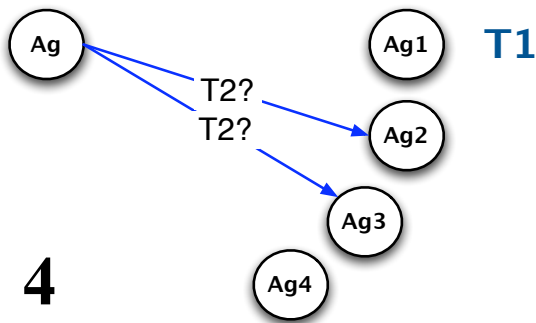
1



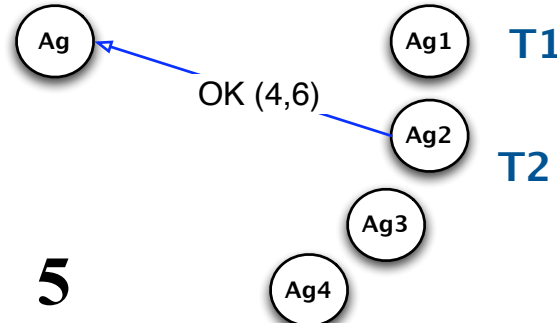
2



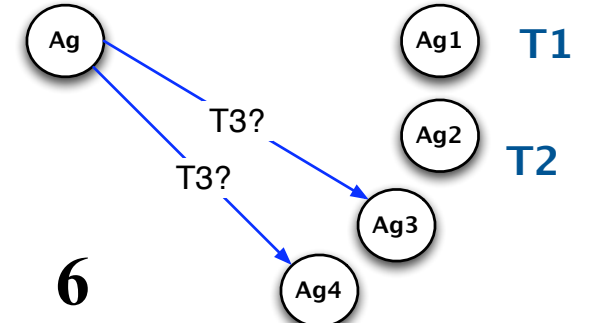
3



4



5

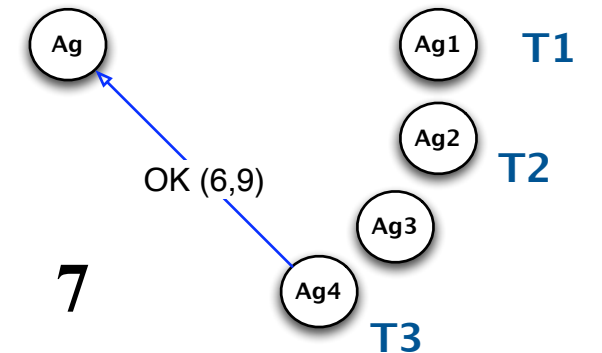


6

T1 before T2
T2 before T3
T3 before instant 10

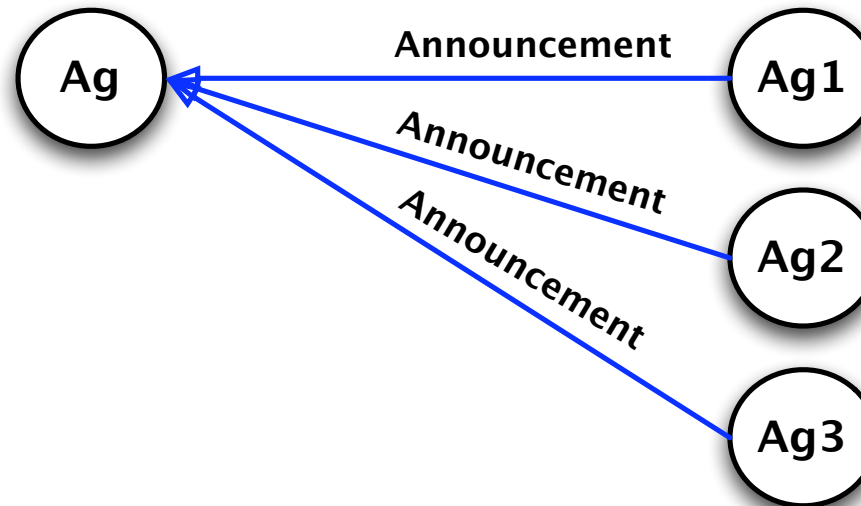
T1 by Ag1 or Ag3, 2 t. units
T2 by Ag2 or Ag3, 2 t. units
T3 by Ag3 or Ag4, 3 t. units

Ag1: [(1,2)]
Ag2: [(3,4)]
Ag3: [(1,3),(5,8)]
Ag4: [(4,5),(9,10)]



7

N to 1 Negotiation



- Combination of other negotiation protocols
- Focused on the contractor's point of view

The potential contractor (Ag) has to decide how it can commit itself with each of the proposals it sends but ...

It is not sure about any of them being accepted by the receivers:

Ag1, Ag2 and Ag3 may reject its proposals or
even being at the same time in a negotiation process with other agents.

Problem:

Suppose that *Ag* receives announcements from

Ag1 to perform the task *T1* with a *3* time units duration to be finished by *instant 4*

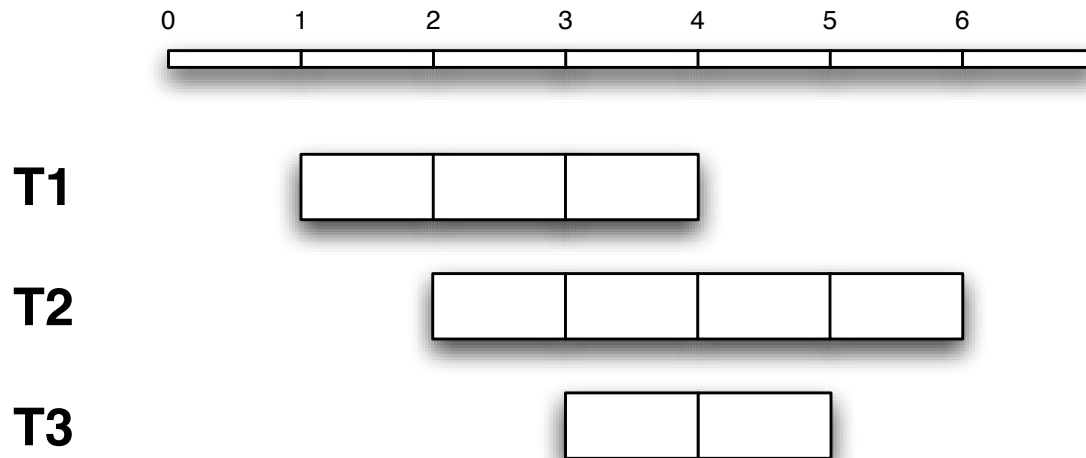
Ag2 to perform the task *T2* with a *4* time units duration to be finished by *instant 6*

Ag3 to perform the task *T3* with a *2* time units duration to be finished by *instant 5*

Ag's agenda is completely free.

Which proposals should *Ag* send to agents *Ag1*, *Ag2* and *Ag3*?

Negotiation Protocols



Ag is be able to perform T1, T2 or T3 and also the pairs T1-T3 or T3-T2.

1. Suppose that Ag chooses T2 because it is longer to execute.
2. Ag will then answer Ag2 and Ag3 with accepting proposals and rejects Ag1 task T1.
3. If Ag2 and Ag3 choose another agent to perform T2 and T3, Ag looses everything (even if Ag1 doesn't have any alternative for T1).
4. Ag made unhappy choices regarding these announcements.
5. If Ag had been less "honest" and accepted all three possibilities, hoping to get at least one, it could be in trouble if by chance everybody chose its proposals.

Indecision Problem

When an agent takes part in the **simultaneous negotiation** of several contracts it cannot know in advance whether or not its proposals will be accepted.

Consequently its behaviour will be affected.

Scenarios

- **Ag optimist and reliable**
 - considers as unavailable what has been object of proposals
 - tend to be unavailable for further negotiations
 - risks losing additional opportunities if its proposals are refused
- **Ag pessimist and unreliable**
 - knows that there is no assurance that its proposals will be accepted
 - in future negotiations considers all that has been included in prior proposals and not yet accepted as still available
 - maximizes opportunities but risks to be discredited

How to deal with the Indecision Problem?

- Negotiate **proposal to proposal** until the end
- Include **intermediate steps** in the negotiation process (**re-confirmations**)
- Evaluation of the **relative importance** of the potential contracts
- Evaluation of **unfulfilled contracts' impact**
- Use of **subcontracting**
- **Renegotiation** of already awarded contracts

Auction is an assets' attribution method based on competition

Characteristics:

- Auction as a negotiation
- Useful for assets without pre-determined value
- Efficient - allots resources to whom values them the most
- Simple and efficient way of establishing the price
- Flexibility and speed
- Bidders determine the price, Seller sets the rules
- Auctioneer as an intermediary



Generic Types:

- Open or with sealed proposals
- Increasing or Decreasing Prices
- Personal Assets or Resale
- Unilateral or bilateral



Auction settings

Private value auctions

- Value of the good depends only on the agent's own preferences.
- Winning bidder will not resell the item, because the value would depend on other agents' valuations.
- Agent is assumed to know exactly its value for the good.

Common value auctions

- agent's value of an item depends entirely on other agents' values of it.

Auction settings

Correlated value auctions

- Agent's value depends partly on its own preferences and partly on others' values.
- If an agent handles a task itself, only agent's local concerns define the cost of handling it. If the agent can subcontract the task, cost depends solely on other agents' valuations.

Time considerations

- **Open auctions** with **many bidders** that take an **extended period** of time to reach a final bid, will tend to a final price very close to the true market value.
- When there are **few bidders** and each bidder is allowed only **few bids**, the process is much quicker, but the final bid will probably not reflect accurately the real market value.

Auctions Basic Taxonomy

according to Vickrey

Auction type	Rules	Closing Price
English (open, oral, ascending)	Seller may set a “reserve” price. Bidding price increases until there are no more bids. Bidders can bid several times.	Higher bid value
Dutch	Seller announces a high asking price. Price is going down until some bidder accepts current price.	Better proposal value (1st bid)
Sealed-bid first-price auction	Bidders submit their proposals secretly. The winner pays the proposed price.	Better proposal value
Sealed-bid second-price auction	Bidders submit their proposals secretly. The winner is the one that offered the higher price but pays the price offered by the second best proposal.	Value of the second best proposal.

English auction

- First-price open-cry
- Open Ascending Process

Mechanism:

- Auctioneer finds estimated market value
- Auctioneer starts auction by announcing the 1st bidding price (reserve price - typically 50% of estimated market value)
- After the opening bid higher bids follow
- When no more bids are made after a certain period of time, the last bidder wins.

English auction

Analysis:

- An agent's strategy is a series of bids as a function of
 - his private value,
 - his prior estimates of other bidder's valuations, and
 - the past bids of others.
- agent's dominant strategy is to always bid a small amount more than the current highest bid, and stop when his private value price is reached.

First-price sealed-bid auction

Each bidder submits one bid without knowing the others' bids. The highest bidder wins the item and pays the amount of his bid.

Analysis:

- An agent's strategy is his bid as a function of his private value and prior beliefs of others' valuations.
- There is no dominant strategy. The best one is to bid less than his true valuation - how much less depends on what the others bid. The agent should bid the lowest amount that still wins the auction and does not exceed his valuation.

Weiss, 99

Dutch (descending) auction

The seller continuously lowers the price until one of the bidders takes the item at the current price.

Analysis:

- Strategically equivalent to the first-price sealed-bid auction, because in both, an agent's bid matters only if it is the highest, and no relevant information is revealed during the process.
- Dutch auctions are efficient in real time because the auctioneer can decrease the price at a quick pace.

Vickrey (second-price sealed-bid) auction

Each bidder submits one bid without knowing the others' bids. The highest bidder wins, but at the price of the second highest bid.

Analysis:

An agent's strategy is his bid as a function of his private value and prior beliefs of others' valuations.

Theorem

A bidder's dominant strategy in a private value Vickrey auction is to bid his true valuation.

Weiss, 99

Vickrey Theorem explanation

- If he bids more than his valuation, and the increment made the difference between winning or not, he may end up with a loss if he wins.
- If he bids less, there is a smaller chance of winning, but the winning price is unaffected (2nd highest).
- It means then that an agent is best off bidding truthfully no matter what the other bidders are like (capabilities, environments, bidding plans).
- This has two desirable consequences:
 1. agents reveal their preferences truthfully which allows globally efficient decisions to be made.
 2. agents need not waste effort in guessing other agents' bids because they are not relevant to the bidding decision.

Applications

- Vickrey auctions have been widely advocated and adopted for use in computational multiagent systems, namely:
 - to allocate computation resources in operating systems,
 - to allocate bandwidth in computer networks,
 - Google AdWords
 - E-Bay proxy bidding variant
- Vickrey auctions have not been widely adopted in auctions among humans. Maybe because it is counter-intuitive?

Lies and collusion

How immune are these mechanisms?

Bidders

- No mechanism is collusion-proof
- Solution: anonymize! Doesn't work in open-cry methods unless computerized.

Auctioneers

- Vickrey prone to auctioneer lying
- Solutions: bid signing or third-party bid handling
- Use of shills to inflate bidding price in English auctions
- Auctioneer bidding

Conflict of **Goals**

- Different goals, eventually contradictory
- Devising compromises
- Using priorities when compromise is impossible

Conflict of **Responsability** or Interest

- Common goals, but competing for the same task or asset

Conflicts of **Information** or Knowledge

- Different views of the same reality
- Definition of different levels of credibility
- Eventual fusion of the different results
- Eventual use of uncertainty factors

Conflict handling

The search for an acceptable solution for the conflict

- **Flight** – effective in certain MAS settings
- **Destruction** – mistakes not recoverable
- **Subservience** – weak form of destruction, reversible. Difficult in MAS
- **Delegation** – to a judge entity
- **Compromise** – in MAS goals should be modelled as attributes with a certain range
- **Consensus** – completely mutually accepted solution. Difficult to implement in MAS also for privacy reasons.

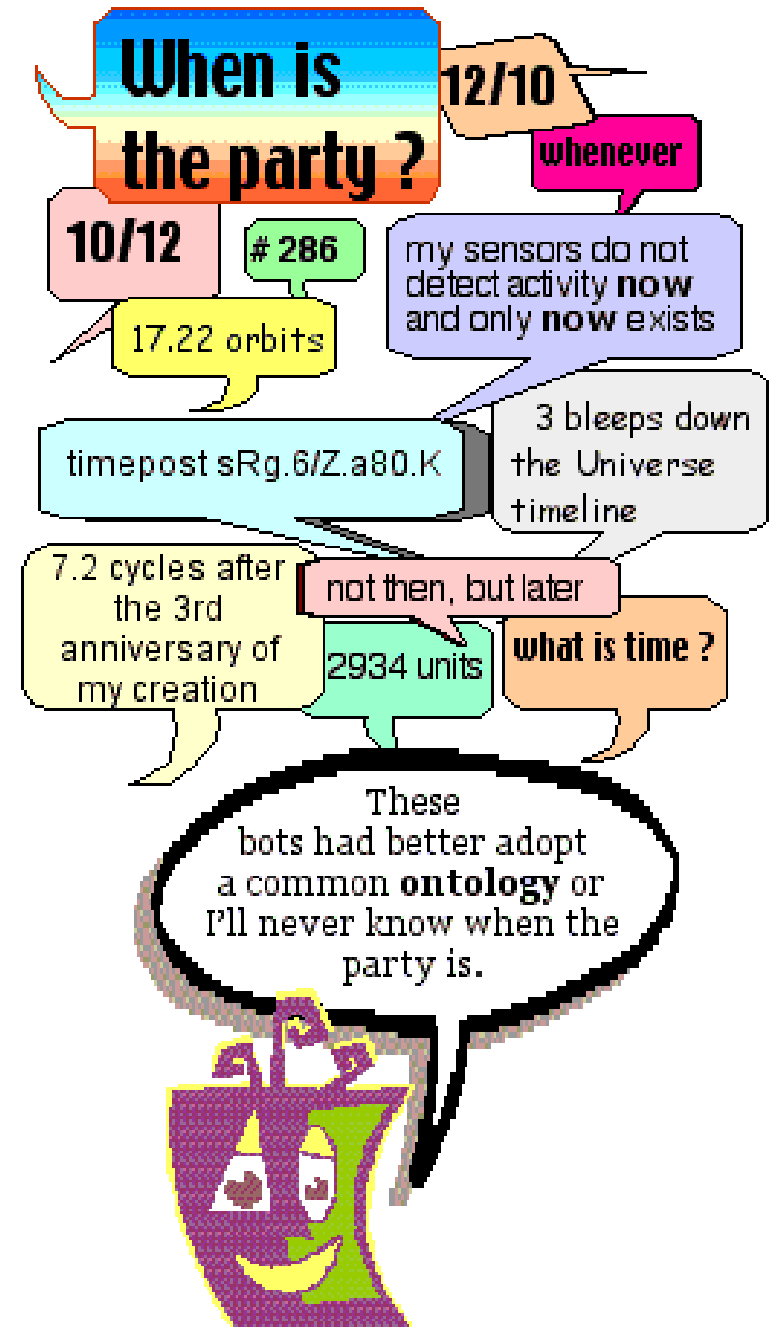
[Muller, 01]

Definitions

An ontology is an explicit specification of a conceptualization

Gruber, 1993

Conceptualization is an abstract, simplified view of a certain domain to be represented.



An ontology is a formal definition of a body of knowledge. The most typical type of ontology used in building agents involves a **structural** component, essentially a taxonomy of class and subclass relations coupled with definitions of the relationships between these things.

Hendler,2000

In the context of multi-agent systems, **ontology is a computer-readable description of knowledge about the resources** ... The software agents become intelligent because they can make use of the knowledge contained in ontology to use in the process of negotiation and decision-making.

Howarth, 2004

The existence of a **common language** is not enough for the agents to understand themselves

Agents must **share** the same knowledge organisation

An ontology includes:
a common **vocabulary**
+
concepts and its **relationships**

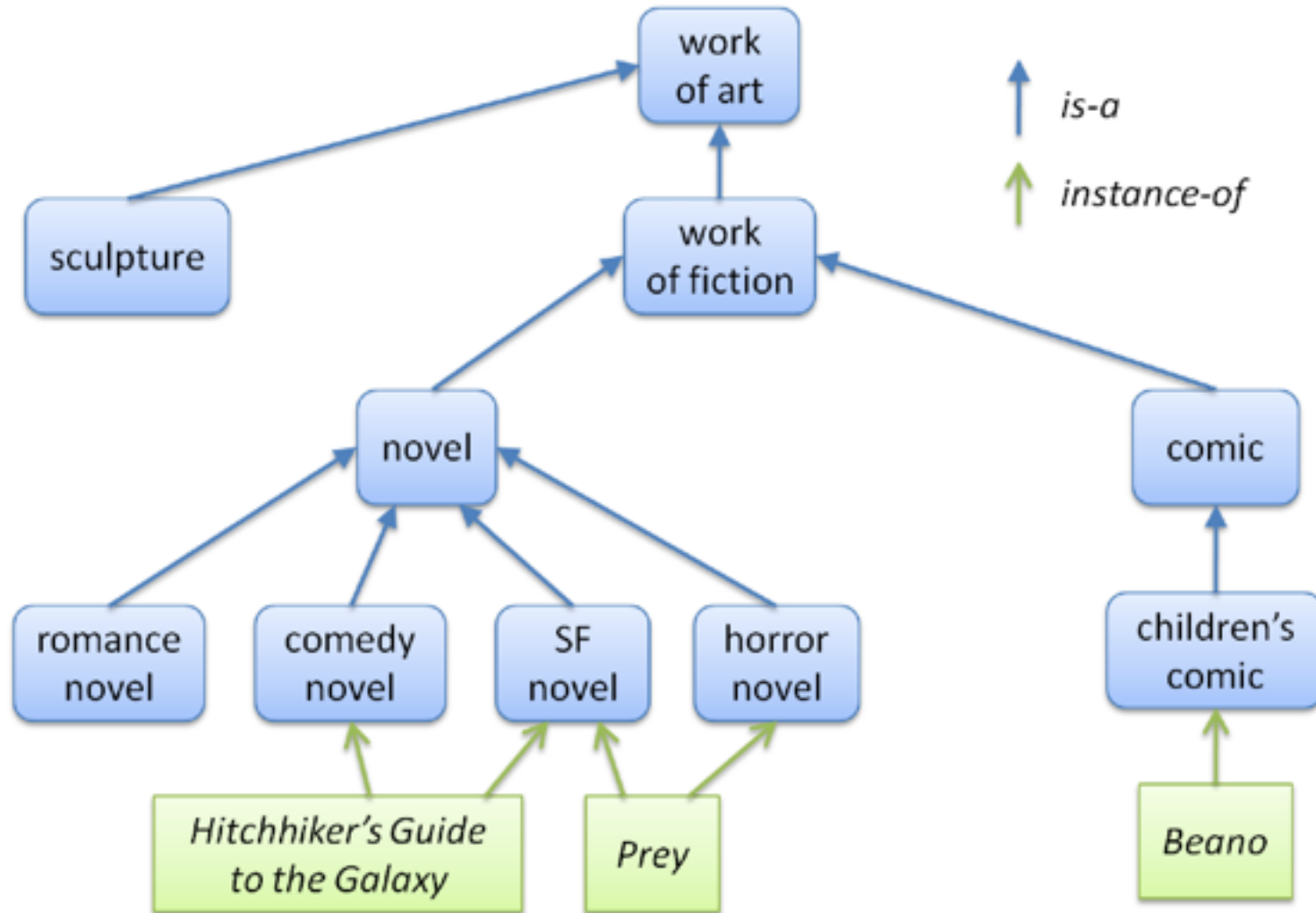
Ontologies are essential for the establishment of a **knowledge exchange** common **platform**. Without it, each one is bound to assign different meanings to the same terms.

Ontological Commitment

Agreement between a group of Agents for the use of a common vocabulary.

An ontology may be represented by a **hierarchical Knowledge Base** structured in **classes**.

Agents Interaction - Ontologies



Michael Wooldridge

Criteria

👤 Clarity

- 👤 Objective and complete definitions

👤 Coherence

- 👤 Avoid contradictions

👤 Extensibility

- 👤 Anticipation of future uses for the shared vocabulary
- 👤 Definition of new terms based on the existent ones

👤 Minimization of Logical Commitments

- 👤 An ontology should include the definitions strictly needed for communicating the knowledge

Gruber, 1993

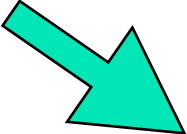
Ontology
Elements
(in SUO-KIF)

```
(subclass Person Animal)
```

```
(and  
  (instance KofiAnnan Human)  
  (occupiesPosition KofiAnnan SecretaryGeneral UnitedNations))
```

```
(not  
  (occupiesPosition SilvioBerlusconi President Libya))
```

```
(=>  
  (and  
    (instance ?P Human)  
    (attribute ?SL Asleep))  
  (not  
    (exists ?ACT  
      (and  
        (instance ?ACT IntentionalProcess)  
        (overlaps ?ACT ?SL)  
        (agent ?ACT ?P))))))
```



“If a person is sleeping he or she cannot perform an intentional action”

Knowledge Interchange Format (KIF) is a formal language for knowledge interchange between different computational programs, written by different programmers, in different periods of time, using different languages.

Genesereth, 1992

- **KIF is *not* intended as a primary language for interaction with human users** (though it can be used for this purpose).
- Different computer systems can interact with their users in whatever forms are most appropriate to their applications.

- **KIF is also *not* intended as an internal representation for knowledge *within* computer systems ...** (though the language can be used for this purpose as well).
- Typically, when a computer system reads a knowledge base in KIF, it converts the data into its own internal form... All computation is done using these internal forms.
- When the computer system needs to communicate with another computer system, it maps its internal data structures into KIF.

KIF is not:

- a language for user interaction
- a form of knowledge internal representation

KIF:

- is a language with **declarative semantics**
- allow expressing sentences in **1st order logic**
- allows the representation of **meta-knowledge**

Examples

Data Structures

(salary 123456789 accounting 1500)

(salary 132547698 purchasing 1200)

(salary 143276597 marketing 1800)

Expressions

(= (temperature m1) (scalar 83 Celsius))

(> (* (width t1) (length t1)) (* (width t2) (length t2)))

(=> (and (real-number ?x) (even-number ?n)) (> (expt ?x ?n) 0))

Examples

Definitions

```
(defrelation solteiro (?x) :=  
  (and (homem ?x) (not casado ?x))))
```

Meta-knowledge

```
(interested joe ' (salary ,?x ,?y ,?z ))
```

Scripts

```
(progn (fresh-line t) (print "Hello") (fresh-line t))
```

Common logic (CL) is a framework for a family of logic-based languages with the purpose of standardizing syntax and semantics for information exchange.

Although a work in progress, there are already three syntaxes standardized:

- **CLIF** - Common Logic Interchange Format, based on **KIF**
- **CGIF** - Conceptual Graph Interchange Format
- **XCL** - eXtended Common Logic Markup Language, based on **XML**

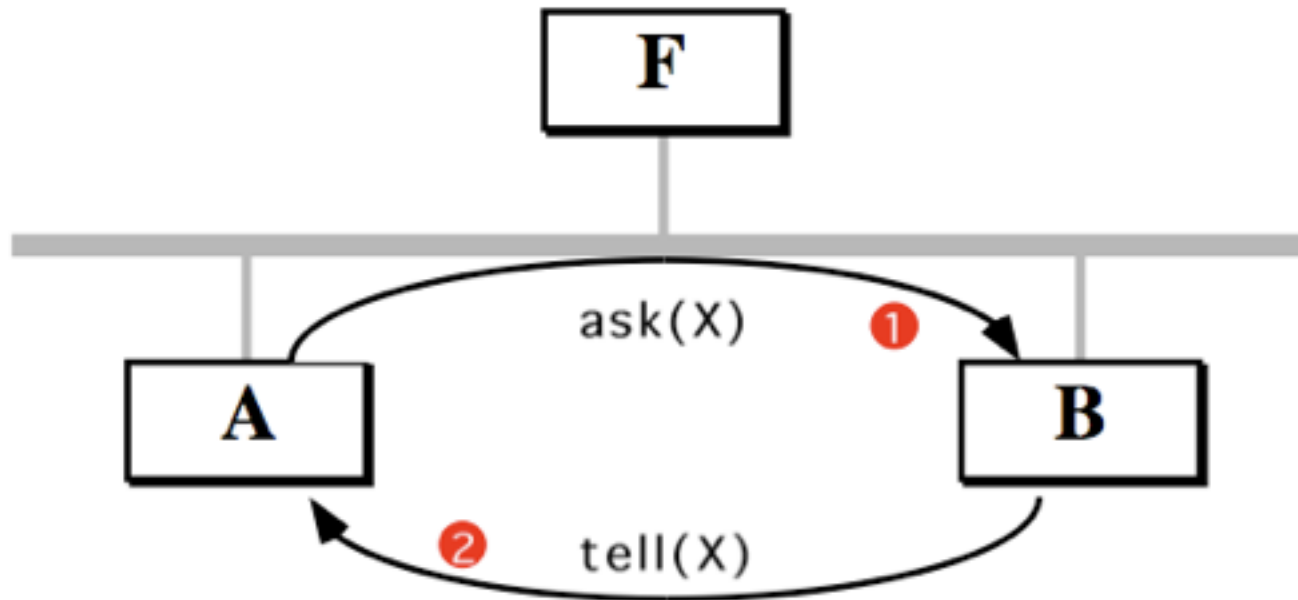
Any statements in any of these languages can be translated to any other language while preserving the original semantics.

KQML (Knowledge Query and Manipulation language) is a language and a **protocol for the exchange of information** and knowledge between agents.

KQML is **indifferent** to the **contents** and **format** of the information that it carries.

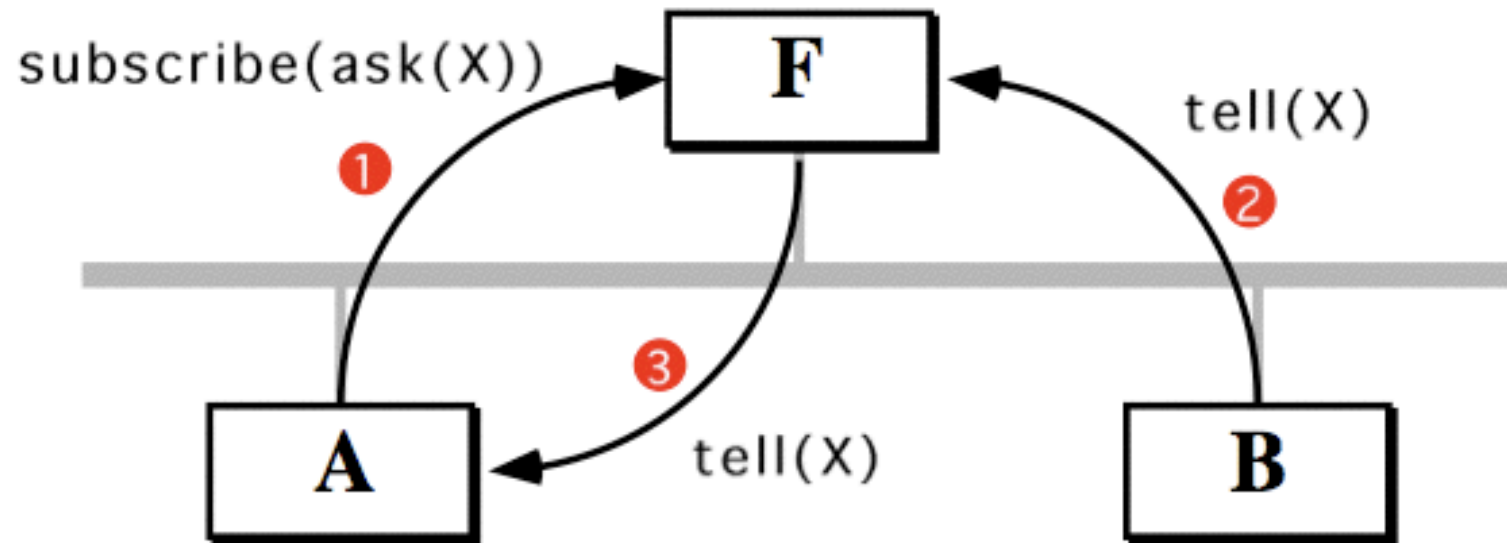
In a society of agents using **KQML** there are usually special agents (**facilitators**) offering services as:

- Association between **physical** and **symbolic addresses**
- **Register** of **databases** or **services**
- **Communication** services like
 - Message forwarding
 - Information brokering
 - Content-based message routing

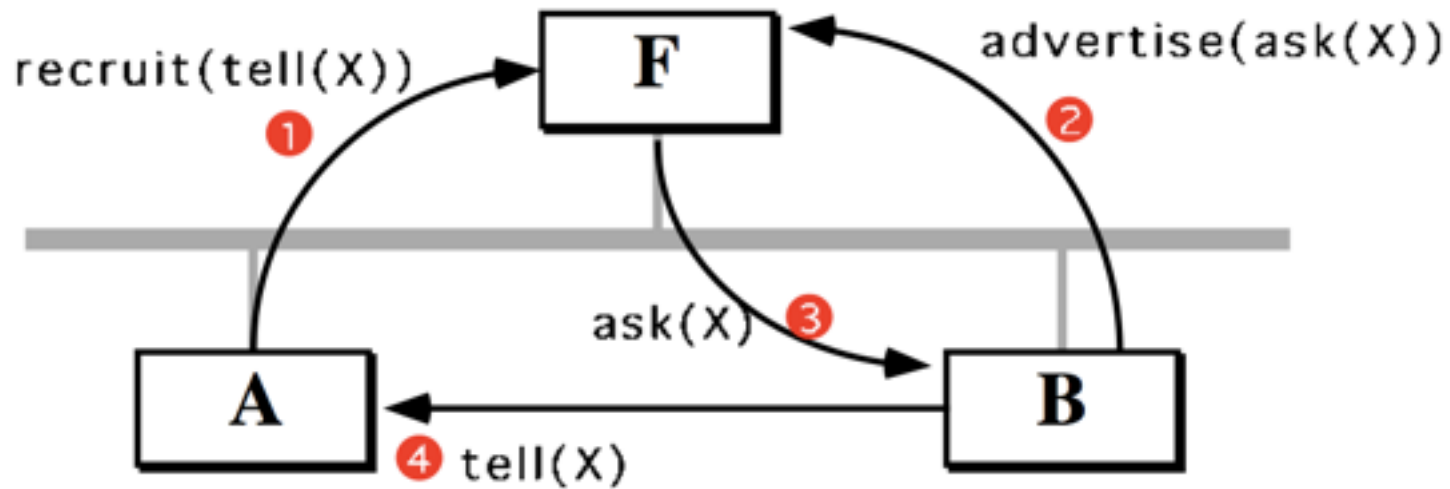


As A is aware of B and of the appropriateness of querying B about X, a simple point-to-point protocol may be used.

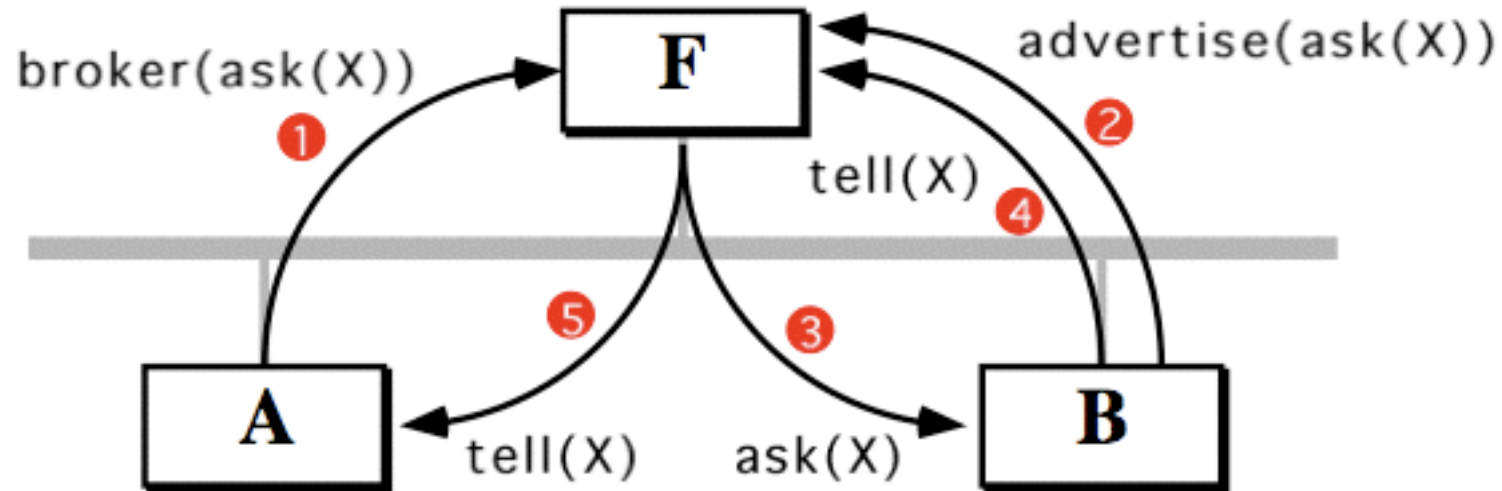
[Finnin, 94]



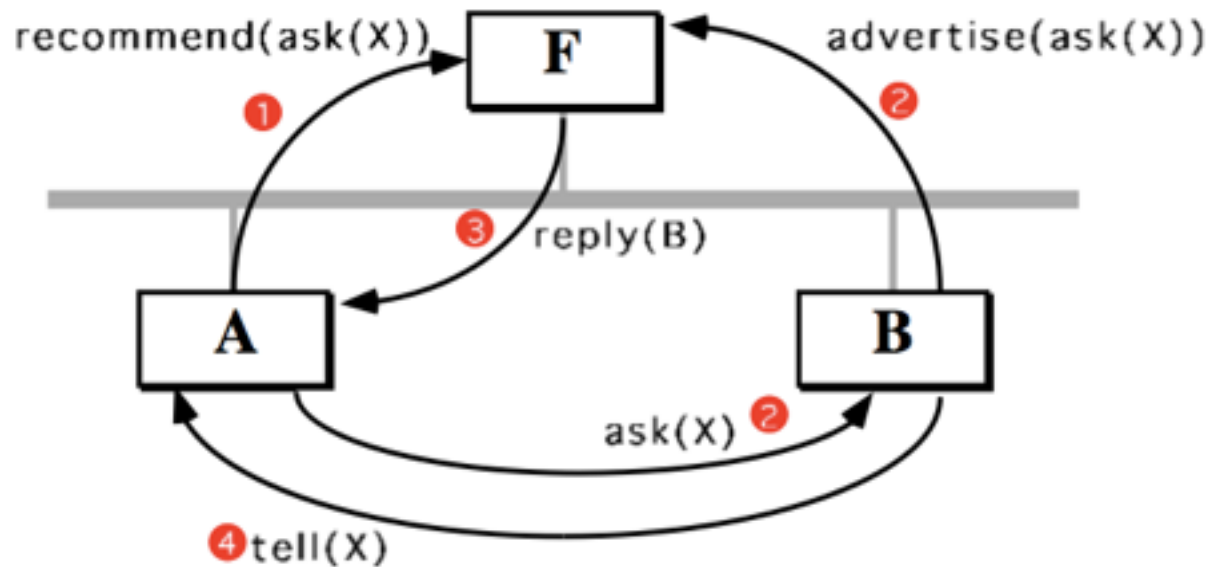
Agent A can ask facilitator F to monitor for changes in its knowledge base (**subscribe** performative).



Using the **recruit** performative, A asks F to find an agent willing to process an embedded performative and send the answer directly to A.



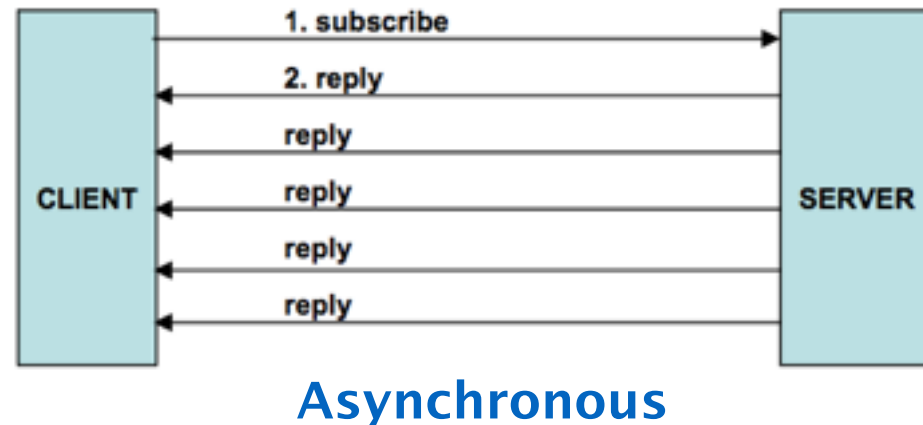
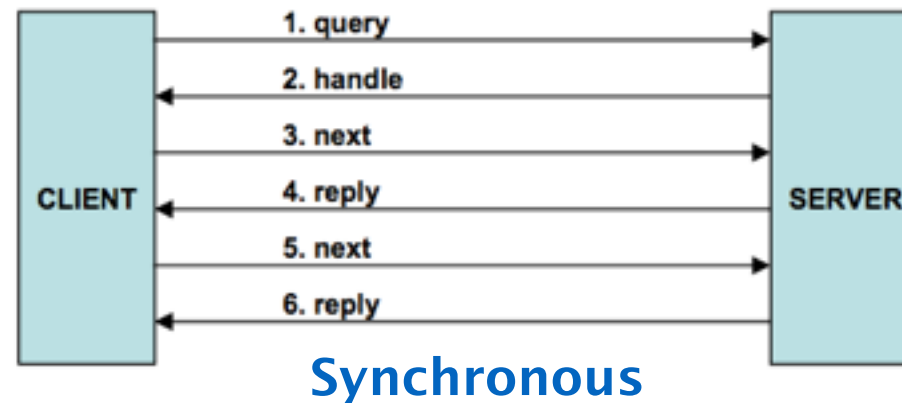
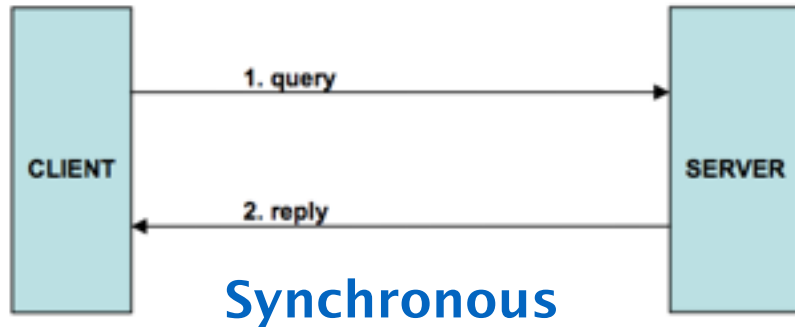
The **broker** performative is used to ask a facilitator to find another agent capable of processing the **ask** performative and to forward the reply.



A asks F to **recommend** an agent willing to accept **ask(X)** performatives. Once F learns about B willingness, sends A this information. All further interaction between A and B is managed by A.

Protocols

[Subramaniam, 2002]



Language primitives are called **performatives**, defining the actions that are allowed in the communications between agents.

(KQML-performative

:sender	<word>
:receiver	<word>
:language	<word>
:ontology	<word>
:content	<word>
...)	

- KQML performative semantics is domain independent
- Message semantics is specified by **:content**, **:language** and **:ontology** fields

Fields of a KQML message

Element	Description
performative	Action performed by the message
sender	Message initiator
receiver	Message recipient
reply-to	Recipient of the message reply
content	Message content
language	Language used to express content
encoding	Encoding used for content
ontology	Ontology context for content
protocol	Protocol message belongs to
conversation-id	Conversation message belongs to
reply-with	Reply with this expression
in-reply-to	Action to which this is a reply
reply-by	Time to receive reply by

Basic query performatives

- evaluate** S wants R to simplify the sentence
- ask-if** S wants to know if the sentence is in R's KB
- ask-about** S wants all relevant sentences in R's KB
- ask-one** S wants one of R's answers to a question
- ask-all** S wants all of R's answers to a question

Multi-response query performatives

- stream-about** multiple response version of ask-about
- stream-all** multiple response version of ask-all

Response performatives

- reply** communicates an expected reply
- sorry** S cannot provide a more informative reply

Generator performatives

- standby** S wants R to be ready to respond to a performative
- ready** S is ready to respond to R's previously mentioned performative
- next** S wants R's next response to a previously mentioned performative
- rest** S wants R's remaining responses to a previously mentioned performative
- discard** S will not want R's remaining responses to a previous performative
- generator** same as a standby for stream-all

Networking performatives

- register** S can deliver performatives to some named agent
- unregister** a deny of a register
- forward** S wants R to route a performative
- broadcast** S wants R to send a performative over all connections

Generic informational performatives

- tell** the sentence is in S's KB
- deny** embedded performative does not apply to S (anymore)
- untell** sentence is not in S's KB.

Capability-definition performatives

- advertise** S is suited to processing a performative
- subscribe** S wants updates to R's response to a performative
- monitor** S wants updates to R's response to a stream-all

Other performatives

- achieve** S wants R to make something true on their environment
- broker-one** S wants R to collect all responses to a performative
- broker-all** S wants R to get help in responding to a performative

Agent Joe asks agent Stock-server about IBM shares' value:

(ask-one ← Performative
 :sender joe
 :content (PRICE IBM ?price) ← Content
 :receiver stock-server ← Receiver
 :reply-with ibm-stock
 :language LPROLOG
 :ontology NYSE-TICKS) ← Ontology

Attribute-Value
pairs

(tell ← Performative
 :sender stock-server
 :content (PRICE IBM 14) ← Content
 :receiver joe ← Receiver
 :in-reply-to ibm-stock
 :language LPROLOG
 :ontology NYSE-TICKS) ← Ontology

Examples of KQML message exchanges

Agent A asks agent B a simple query and receives a response via a tell

Agent A sends the following performative to agent B:

```
(evaluate
  :language KIF
  :ontology motors
  :reply-with q1
  :content (val (torque motor1) (sim-time 5)))
```

and agent B replies with

```
(reply
  :language KIF
  :ontology motors
  :in-reply-to q1
  :content (scalar 12 kgf))
```

[Alan Bond, 2001]

Agent A sends the following performative to agent B:

```
(stream-about  
  :language KIF  
  :ontology motors  
  :reply-with ql  
  :content motor1)
```

Agent A asks agent B to tell all it knows about motor1.

Agent B replies with a sequence of tells terminated with a sorry

and agent B replies with a series of performatives:

```
(tell  
  :language KIF  
  :ontology motors  
  :in-reply-to ql  
  :content (= (val (torque motor1) (sim-time 5) (scalar 12 kgf)))
```

```
(tell  
  :language KIF  
  :ontology structures  
  :in-reply-to ql  
  :content (fastens frame12 motor1))
```

```
(sorry :in-repl-to ql)
```

Agent A tells Agent B to achieve a state in which the torque of motor1 is a particular value

Agent A sends the following performative to agent B:

(**achieve**

:language KIF

:ontology motors

:reply-with q1

:content (= (val (torque motor1) (sim-time 5)) (scalar 2 kgf))

and after achieving the requested motor torque, agent B might send the following (though it is not mandatory):

(**tell**

:language KIF

:ontology motors

:content (== (val (torque motor1) (sim-time 5)) (scalar 2 kgf)))

- Agent A asks B to prepare to generate a stream of all of the information if knows about motor1.
- Agent B replies that it is ready and returns an identifier for A to use in requesting the individual facts.
- Agent A asks for a number of facts and finally indicates that no more are required.

Agent A sends the following performative to agent B:

```
(standby
  :language KQML
  :ontology KI0
  :reply-with g1
  :content (stream-about :language KIF :ontology motors
                  :reply-with q3 :content motor1))
```

and agent B replies with:

```
(ready
  :reply-with 2FOB :in-reply-to g1)
```

cont.:

then agent A follows with:

(**next**
:in-reply-to 2FOB)

to which agent B replies with:

(**tell**
:language KIF
:ontology motors
:in-reply-to q3
:content (== (val (torque motor I) (sim-time 5)) (scalar kgf))

and so on, until agent A sends:

(**discard**
:in-reply-to 2FOB)

KQML messages can be nested

If Agent 1 cannot communicate with Agent 2 it may ask Agent 3 to forward the message to Agent 2:

(forward

:from Agent 1
:to Agent 2
:sender Agent 1
:receiver Agent 3
:language KQML
:ontology kqml-ontology
:content (tell

:sender Agent 1
:receiver Agent 2
:language KIF
:ontology Blocks-World
:content (On (Block A)(BlockB))))

[Weiss, 99]

KQML – Critical analysis

- Basic KQML performative set too large and **not standardized** – different incompatible implementations of KQML
- The language miss the **commissive** performatives, key to agent coordination
- These and other criticisms led to the development of a new ACL by the **FIPA** consortium
- FIPA–ACL was officially accepted by the IEEE in 2005

[Kleiner, Nebel]

KQML-like syntax

```
(inform  
  :sender agent1  
  :receiver agent2  
  :content (price good2 150)  
  :language sl  
  :ontology hpl-auction  
)
```

Also similar set of message attributes

List of Performatives

Requesting information

subscribe	sender asks to be notified when statement changes
query-if	direct query for the truth of a statement
query-ref	direct query for the value of an expression

[Kleiner, Nebel]

Requesting information

inform	<ul style="list-style-type: none">• together with request most important performative;• basic mechanism for communicating information;• sender wants recipient to believe info and believes in it itself
inform-ref	informs other agent about value of expression (in :content)
confirm	confirm truth of content (recipient was unsure)
disconfirm	confirm falsity of content (recipient was unsure)

Negotiation

cfp	<ul style="list-style-type: none">• call for proposals;• initiates negotiation between agents;• content-parameter contains action (e.g.: “sell me car”) and condition (e.g.: “price < 1000\$”)
propose	make proposal
accept-proposal	sender accepts proposal made by other agent
reject-proposal	sender does not accept proposal

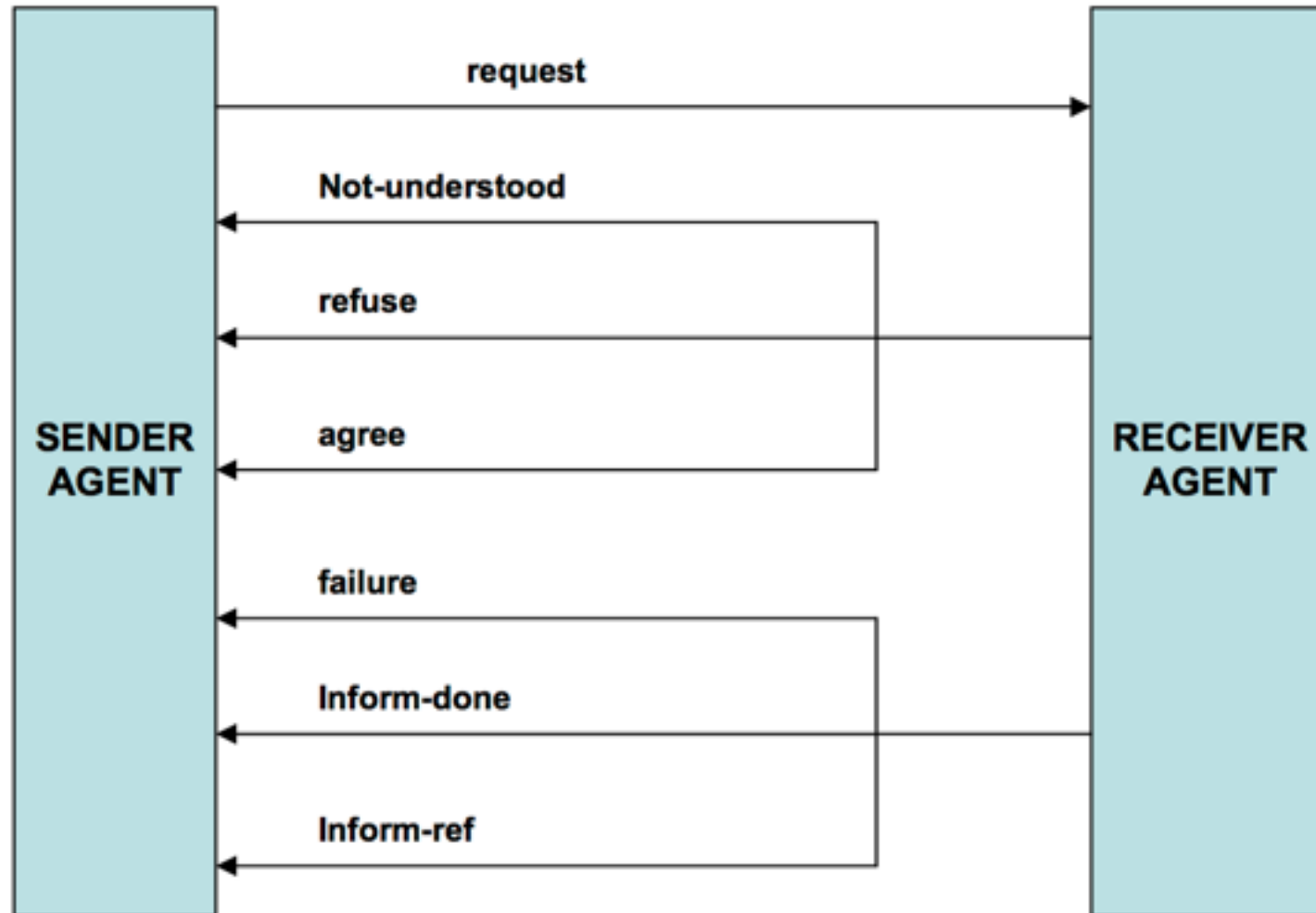
Performing actions

request	issue request for an action
request-when	issue request to do action if and when a statement is true
request-whenever	issue request to do action if and whenever a statement is true
agree	sender agrees to carry out requested action
cancel	follows request; indicates intention behind request is not valid any more
refuse	reject request

FIPA Interaction Protocols
standard exchanges of
performatives according to
well defined situations.

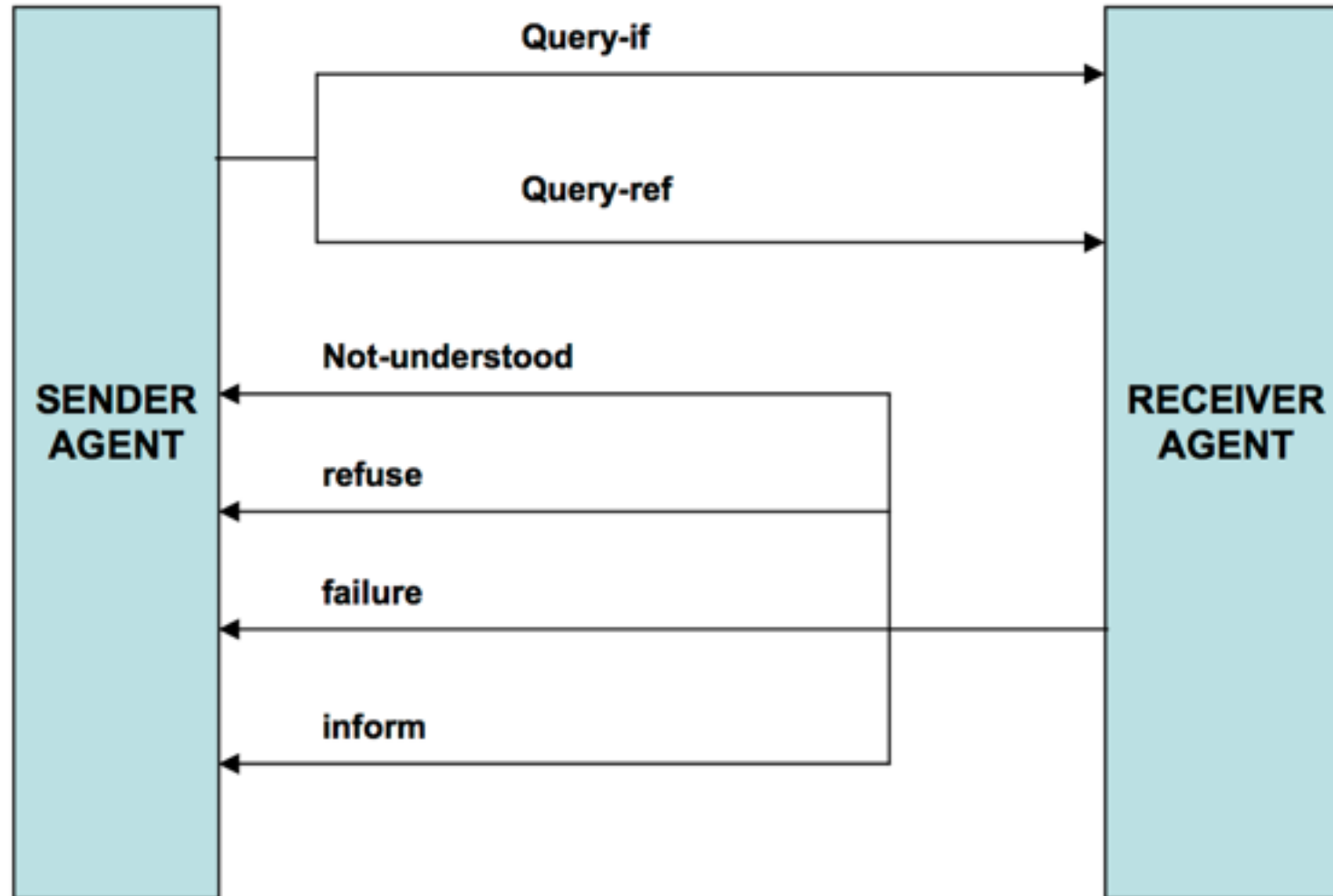
FIPARequest
FIPAQuery
FIPARequestWhen
FIPAContractNet
FIPAIteratedContractNet
FIPAAuctionEnglish
FIPAAuctionDutch
FIPABrokering
FIPAREcruiting
FIPASubscribe
FIPAPropose

FIPA Request Interaction Protocol



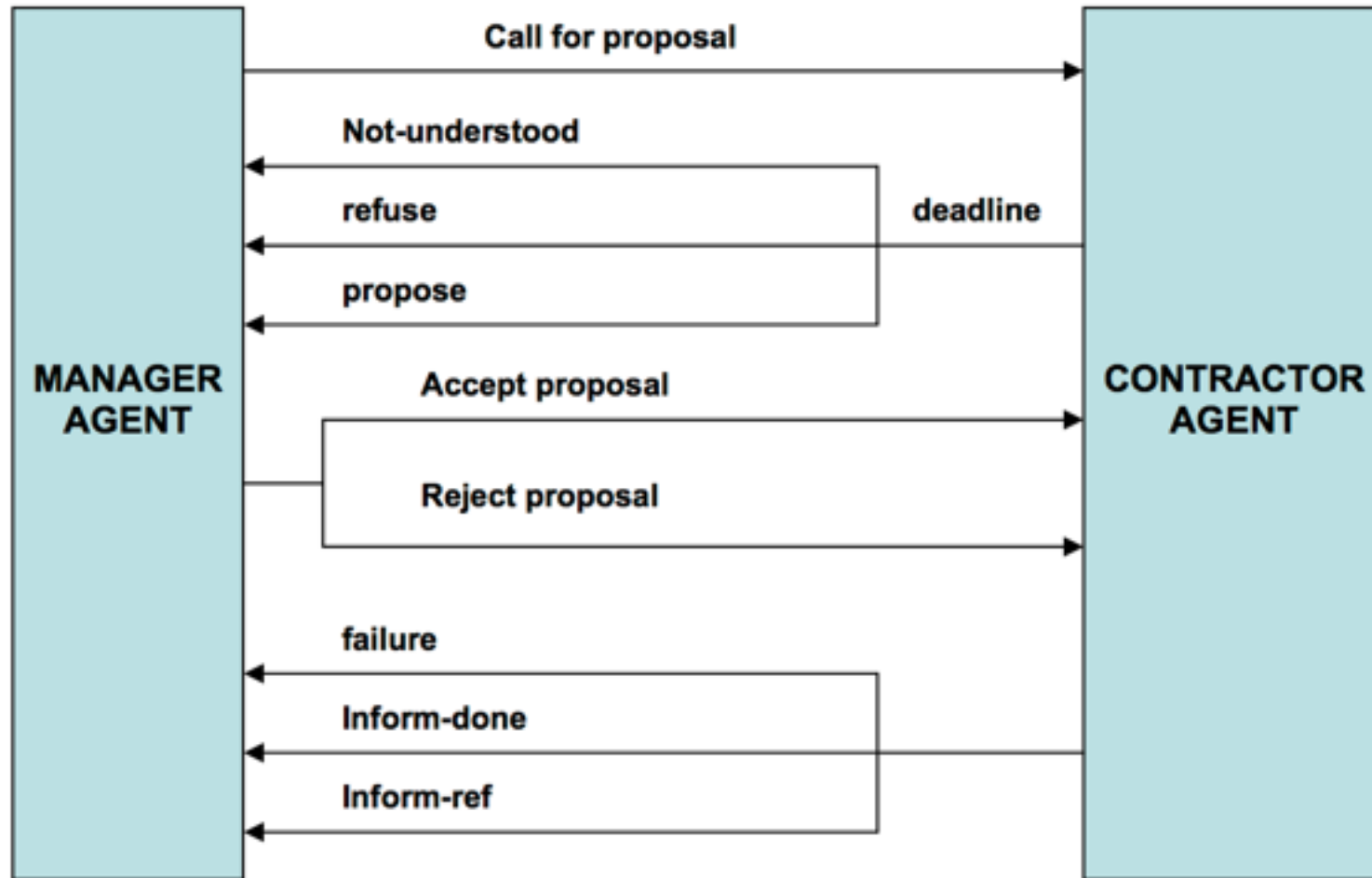
[Subramaniam, 02]

FIPA Query Interaction Protocol



[Subramaniam, 02]

FIPA Contract Net Interaction Protocol



[Subramaniam, 02]

An **ACL (Agent Communicaton Language)** includes 3 components:

- ▶ its vocabulary
- ▶ its *internal* language (KIF)
- ▶ its *external* language (KQML-like)

More than exchanging messages, the agents engage in conversations

Conversation/Speech Acts

Assertive: the door is closed

Directive: close the door

Query: is the door closed?

Commitment: i will close the door

Permissive: he can close the door

Prohibitive: he cannot close the door

Declarative: this is the main door

Expressive: i would like that this was the main door

- **AgentBuilder**
- **AgentTalk, NTT**
- **Agent Toolkit (Win-Prolog)**
- **Aglets, IBM/Japão**
- **JAFMAS**
- **JATLite**
- **JINI**
- **Open Agent Architecture**
- **Repast**
- **Swarm**
- **Voyager**
- **ZEUS, British Telecom**
- **JADE**

A software methodology is characterized by

a **modeling language**,
used for the description of models,

+

a **software process**,
defining the development activities,
its relationships, and how they are
performed.

[Luck, 04]

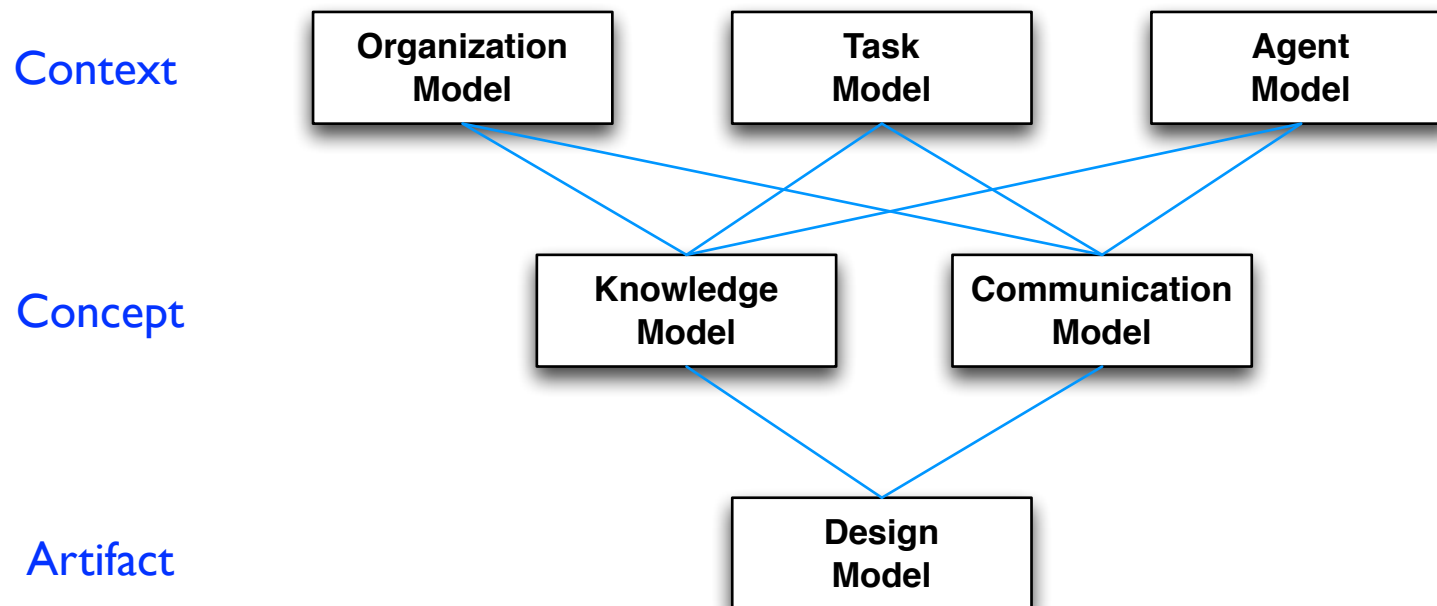
In agent-based software development, different approaches:

- Knowledge engineering
- Agent-oriented
- Extensions to OO

[Luck, 04]

Knowledge Engineering approach

- CommonKADS developed to support knowledge engineers in modeling expert knowledge
- Agent-specific extensions to CommonKADS (CoMoMAS, MAS-CommonKADS)



[Luck, 04]

CommonKADS Models

Knowledge Engineering approach

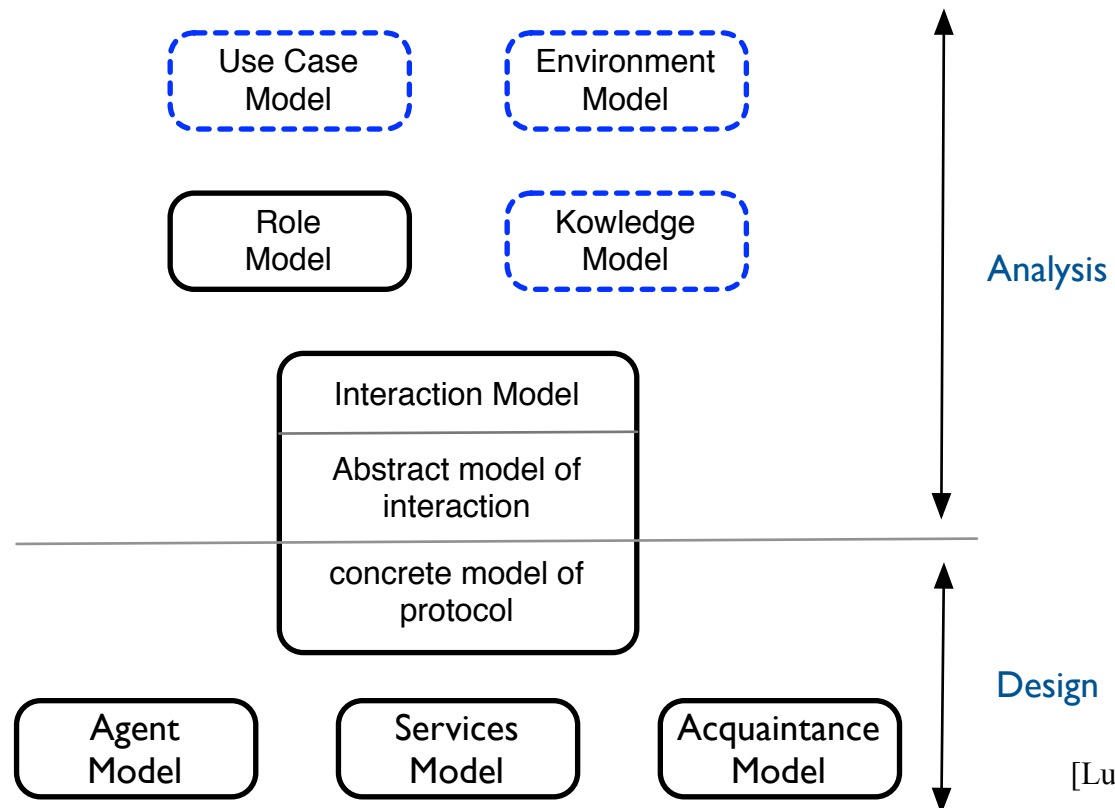
- **Organization** – identifies knowledge providers, users and decision makers
- **Task** – describes tasks performed by agents, with well-defined input and output, goals and constraints
- **Agent** – describes roles & capabilities of agents capable of performing task model's tasks. MAS-CommonKADS adds services, goals, skills, languages and constraints
- **Knowledge** – describes knowledge intensive problem-solving capabilities (domain, inference, task and strategic knowledge)
- **Communication** – transactions. MAS-CommonKADS adds coordination model, including conversations and its protocols
- **Design** – describes application, architectural and platform design stages

[Luck, 04]

GAIA / ROADMAP

proposed by Wooldridge, Jennings & Kinny

- Methodology for agent-oriented analysis and design supporting macro (societal) and micro (agent) levels
- ROADMAP adds requirement analysis



[Luck, 04]

GAIA / ROADMAP

Analysis phase models

- **Use case [Roadmap]** – similar to UML
- **Environment [Roadmap]** – tree hierarchy of zones, described by attributes: static objects, objects, constraints, sources of uncertainty and assumptions.
- **Knowledge [Roadmap]** – domain knowledge assigned to specific roles
- **Role** – responsibilities, permissions, activities and protocols
- **Interaction** – rules of interaction between roles

GAIA / ROADMAP

Design phase models

- **Interaction** – between different roles
- **Agent** – agent types (a set of agent roles)
- **Services** – agent's functions characterised by input, output, preconditions and postconditions, needed to perform agent's role
- **Acquaintance** – lines of communication between different agents

Gaia targeted applications

1. Agents are **coarse-grained** computational systems, each making use of significant computational resources;
2. It is assumed that the goal is to obtain a system that maximises some **global quality measure**;
3. Agents are **heterogeneous**, no assumptions are made about the delivery platform;
4. The **organisation structure** of the system is **static**;
5. The **abilities** of agents and the services they provide are **static**;
6. The overall system contains a comparatively small number of **different agent types**.

Agent Modeling for BDI Agents

proposed by Kinny & Georgeff

Methodology for external modelling

1. Identify the **roles** of the application domain and their lifetimes. Create a **agent class hierarchy**.
2. For each role, identify **responsibilities** and **services** provided.
3. For each service, identify the **interactions** needed, the **performatives** used and their content.
4. Refine agent hierarchy, composing new agent classes, via inheritance or aggregation, guided by commonality of lifetime, information and interfaces, and similarity of services.

[Luck, 04]

Agent Modeling for BDI Agents

Methodology for internal modelling

1. **Analyse means to achieve goals:** for each goal analyse different **contexts**, for each context **decompose** goal into activities. Generate **plan** to achieve goal.
2. **Build beliefs of the system.** Analyse various contexts and conditions, and **decompose** them into component beliefs. Analyse **input and output requirements** for each subgoal in plan.

Iterate these steps as the models are refined.

Agent Modeling for BDI Agents

Models for external viewpoint

- **Agent** – describes hierarchical relationship between different
 - **Agent Classes** – similar to UML class diagram denoting abstract and instanciable agent classes and identifies
 - **Agent Instances** – specifies the initial belief and goal states
- **Interaction** – describes agents' responsibilities, services and associated interactions. Specifies control relationships between classes. Includes syntax and semantics for inter-agent communication

Agent Modeling for BDI Agents

Models for internal viewpoint

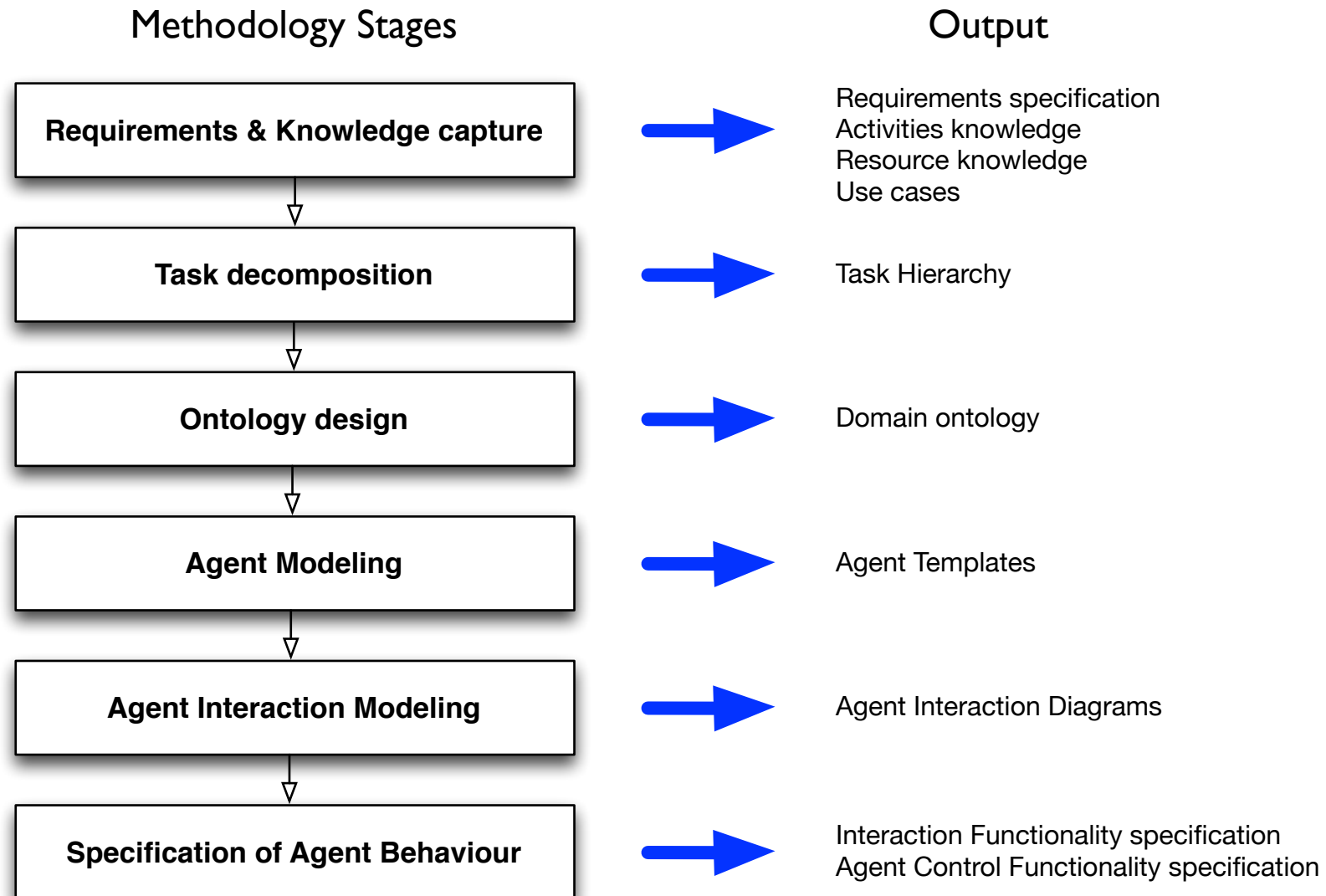
- **Belief** – contains a **belief set** (beliefs and their properties) and one or more **belief states** (instances of the belief set)
- **Goal** – describes the goals an agent may adopt and events to which it may react. Contains a **goal set** and one or more **goal states**. Uses **achieve**, **verify** and **test** modal goal operators.
- **Plan** – consists of a plan set, describing properties and control structure of individual plans.

Agent UML

Extensions to UML covering:

- **Interaction Protocols** – used in FIPA interaction protocols.
- **Social structures** – roles, environments, groups
- **Agent Classes** and roles, state descriptions, actions, methods, capabilities, communicative acts
- **Ontologies**
- **Goals & Plans** – using state charts and activity diagrams

Good part of these extensions already included in UML 2



Agent design methodology used in the design of the PEDDA system

<http://sites.ieee.org/pes-mas/agent-technology/design/>

- **Electronic Commerce**

- Web search for prices: Jango, Bargainfinder
- B2B: FairMarket
- Stock Market: E-Trade, OptiMark
- Auctions: AuctionBot
- Market Simulators: MAGMA, Kasbah, Tête-à-Tête, ISEM

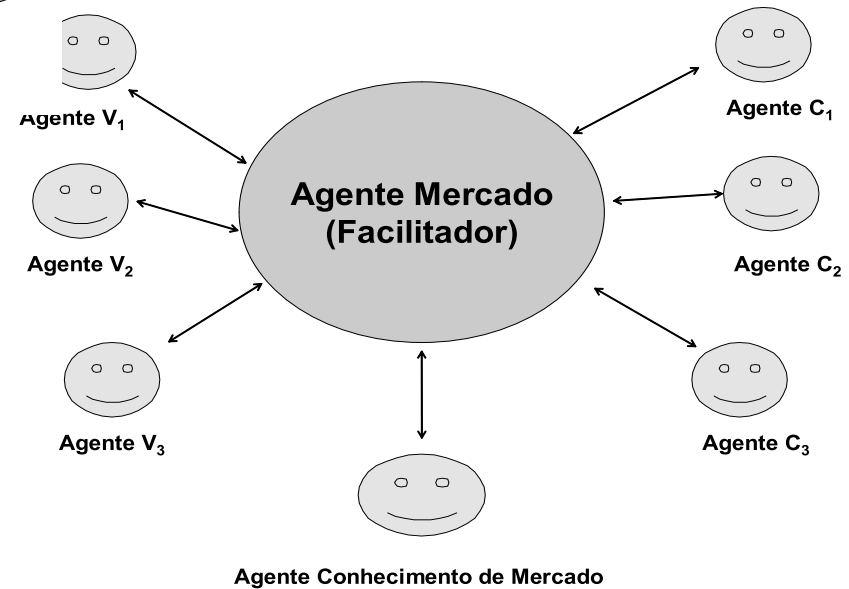
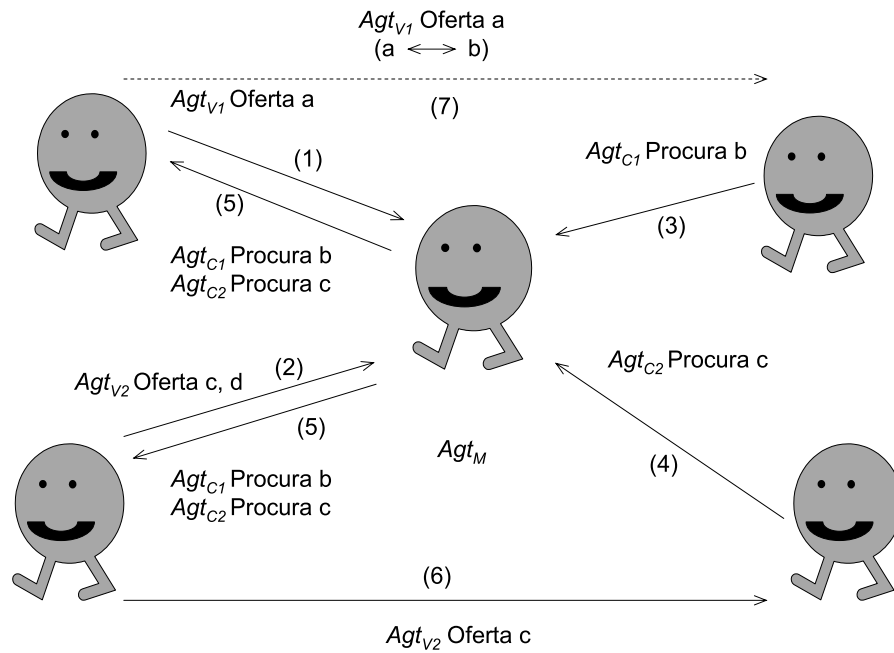
- **Electricity Markets**

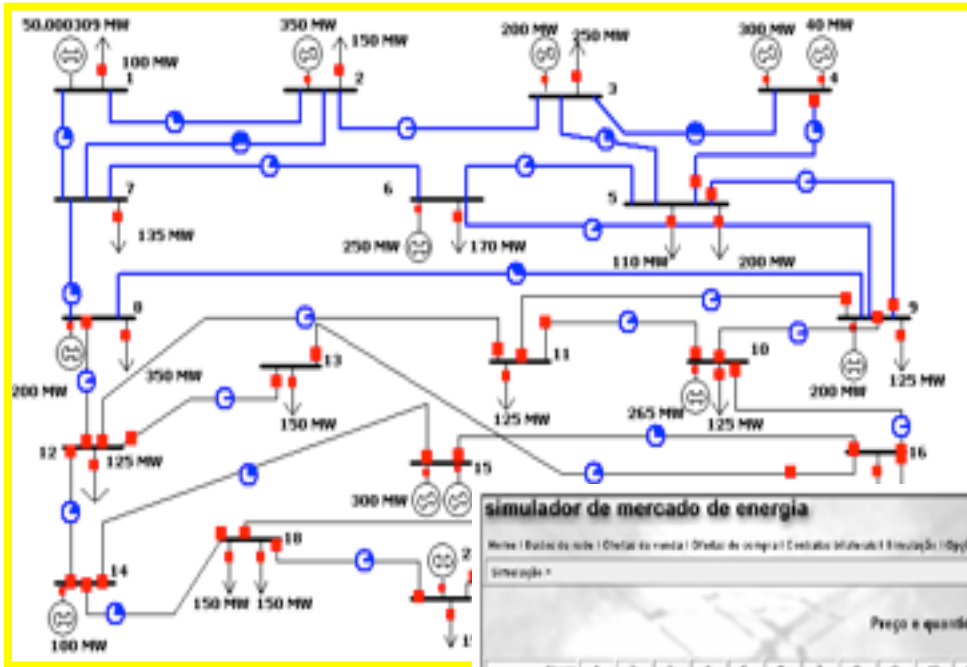
- Electricity Auctions: AMS, AAEPI
- Bilateral Contracts: SEPIA
- Electricity StockMarket: PowerWeb
- Mixed Markets: EMCAS, MASCEM

- **Manufacturing Systems**

- Contract Net based: YAMS, General Electric
- Truck painting: FLAVORS
- Logistics: LMS
- Metallurgy: ADS da Hitachi (Kawasaki Steel)
- ERP+MES: AARIA
- Holonic Systems: HMS, Fabricare

- **Traffic Control Systems**
 - Semaphore Coordination: [DVMT](#)
 - Railways: [Hitachi's ADS](#), for Shinkansen trains
 - Airports: [OASIS](#), Sidney Airport
- **Sistemas Eléctricos de Energia**
 - Incident Analysis and P.S. Restoration: [SPARSE](#), [ARCHON](#), [RESTRAIN](#) (Tutor)
 - Load Management: [Homebot](#)
- **Information Gathering and Filtering**
 - Email filters: [MAXIMS](#)
 - Advise on articles to read: [NEWT](#)
 - Tourist Information: [GALAXY](#)
 - Data organization: [ZDL](#)
 - Image Annotation: [ARIA](#) (KODAK)
- **Space Applications**
 - Planning, Execution and Monitoring: [RemoteAgent](#) (Deep Space 1)
- **Group Decision Making**
 - Argumentation and Emotional Component: [ArgEmotionAgents](#)





simulador de mercado de energia

Menu | Estado do sistema | Oferta de venda | Oferta de compra | Estado do mercado | Simulação | Ajuda

Simulação >

Preço e quantidade total em

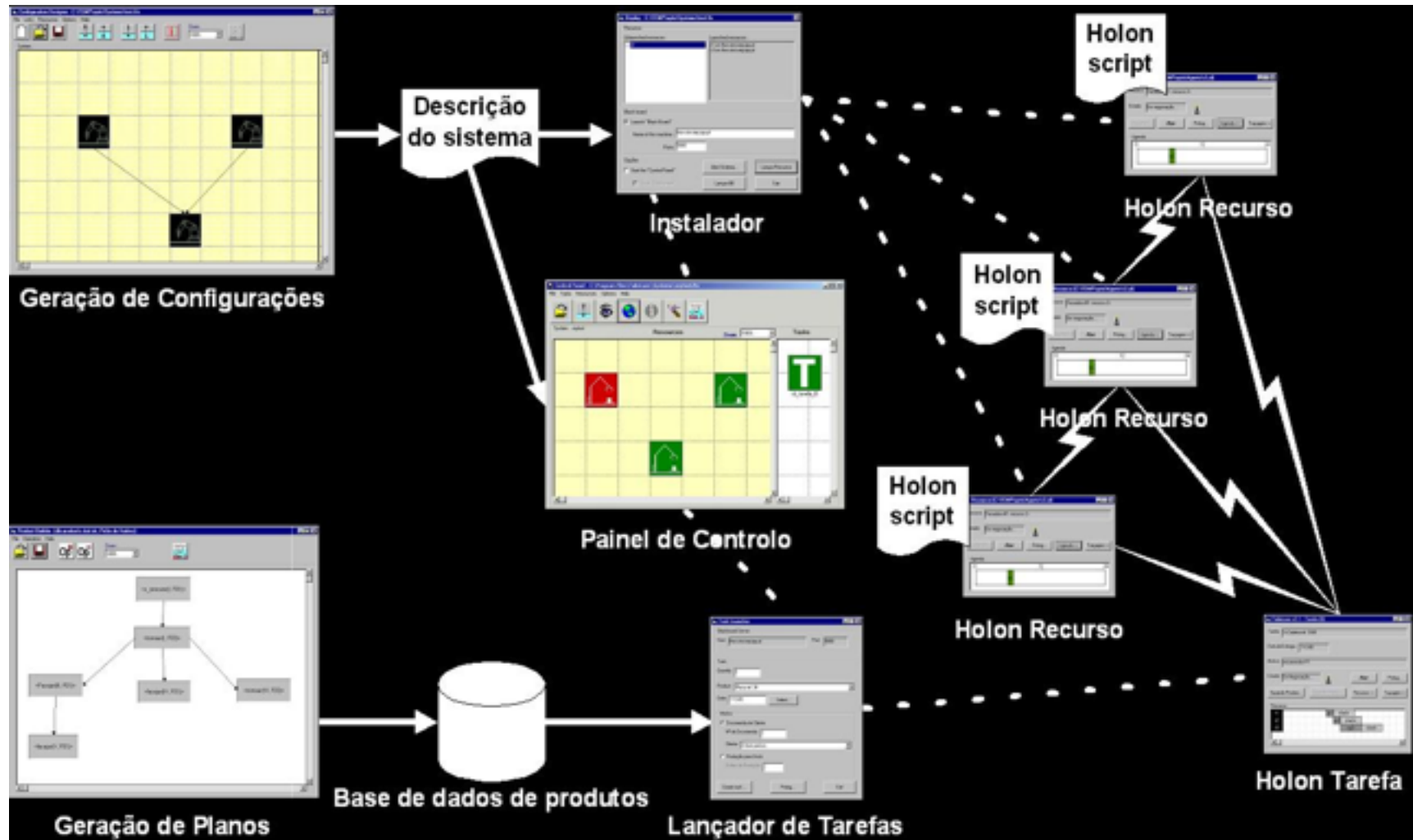
hora	1	2	3	4	5	6	7	8	9	10	11	12	1
Preço \$/MWh	3.6	3.6	3.6	3.4	3.8	3.8	6.8	7.3	11	14	15.8	9.2	1
Energia MWh	240	250	280	280	240	270	230	628	880	798	880	816	4

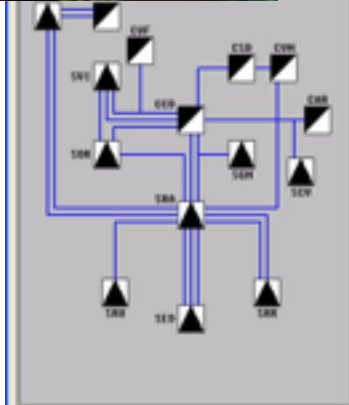
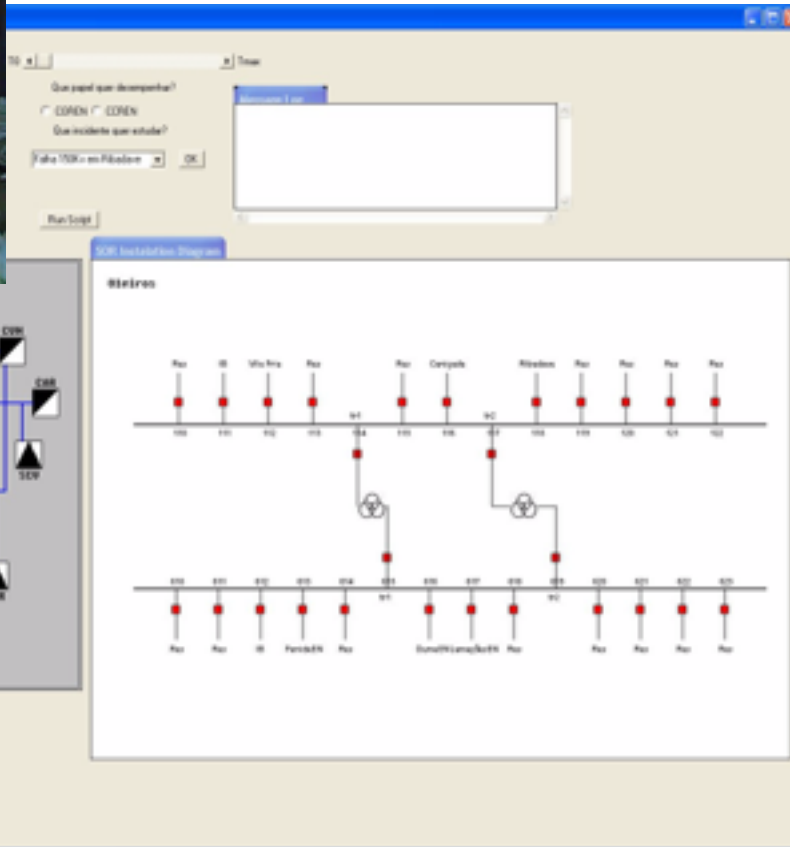


Ofertas de venda aceitas

hora	1	2	3	4	5	6	7	8	9	10	11	12	1
Função	1*												
Vendedor	1	1	1	1	1	1	2	2	2	3	1	1	1
Nº de venda	1	1	1	1	1	1	2	2	2	4	1	1	1
Subida oferta	1	1	1	1	1	1	1	1	1	1	1	1	1
Energia MWh	30	30	30	30	30	30	60	60	140	140	140	140	30
Preço \$/MWh	3.4	3.2	3.1	3	3.2	3.2	5.4	5.8	8.1	11.9	14	9	3.9
Função	2*												
Vendedor	1	1	1	2	2	2	1	2	2	1	2	2	1
Nº de venda	1	1	1	2	2	2	1	2	2	1	2	2	1
Subida oferta	1	2	2	1	1	1	2	2	1	1	2	1	2
Energia MWh	100	80	80	100	80	80	80	80	140	180	130	120	80
Preço \$/MWh	8.5	8.2	8.2	8.1	8.2	8.9	8.8	9.7	8.8	12.8	14.1	8.1	8.1
Função	3*												
Vendedor	2	2	2	1	3	3	2	1	1	3	3	1	1
Nº de venda	2	2	2	1	4	5	2	1	1	4	4	1	4

1 solution(s) found for info(3:3302414_0_8720_8739)_get_satisfiers(_8774)





Conclusions

- * Agents represent a new paradigm in Modeling and Systems development
 - * Agent-based Programming
 - * Multi-Agent Systems
- * Intelligent Agents: concept focused on the agent and the intelligence that it may exhibit
- * Multi-Agent Systems: concept focused on the social skills of a community of Agents
- * Main problems
 - * lack of trust in delegating responsibilities
 - * insufficient maturity on system development
- * A lot of prototypes/systems, some success stories

- ≡ <http://www.agentlink.org/>
- ≡ <http://www.fipa.org/>
- ≡ <http://agents.umbc.edu/>
- ≡ <http://agents.media.mit.edu/>
- ≡ <http://www-2.cs.cmu.edu/~softagents/>
- ≡ <http://agent.cs.dartmouth.edu/>
- ≡ <http://www.psychedelix.com/agents.html>
- ≡ <http://www.cs.umbc.edu/kqml/>

- [Weiss, 99] Gerard Weiss, “Multiagent Systems - A modern approach to Distributed Artificial Intelligence”, MIT Press, 1999
- [Muller, 01] Heinz-Jurgen Muller et al, “Conflicting Agents - Conflict management in Multiagent systems”, Kluwer, 2001
- [Subramaniam, 02] Kalaivani Subramaniam, “Agent Communication Languages and Protocols”, 2002
- [Bond, 01] Alan Bond, <http://www.exso.com/courses/cs101c/kqml/node1.html>, 2001
- [Jennings et al, 1998] Jennings, N.R., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems
- [Finnin, 94] Finnin, Fritzson, KQML as an Agent Communication Language, 1994
- [Luck, 04] Luck, M. Ashri, R., D’Inverno, M., Agent-Based Software development, 2004

[Wooldridge, 00] M. Wooldridge, N. Jennings, D. Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, Kluwer Academic Publishers