SISTEMAS BASEADOS EM AGENTES

AGENT-BASED SYSTEMS

Carlos Ramos
António Silva
DEI-ISEP
2006-2012

1. Introduction

The concept of Agents has come into widespread use lately, hinting to its importance as a programming paradigm, just as in the past Structured Programming and Object-oriented Programming were.

That's why this new technology was first introduced in the AISC (Intelligent Agents and Cooperative Systems) module of the former Computers and Systems branch of Informatics Engineering BsC, followed by this Agent-based Systems module of the Informatics Engineering MsC.

This field's knowledge is spread over a large number of books, magazines and websites. This document's purpose is to try and concentrate in a sole text the most important topics to be taught in this module. Obviously, no document of this type can be complete and, therefore, other complementary texts must be consulted.

Throughout this document we pretend to clarify concepts like Agent or Multi-Agent System, trying to address the conditions under which this new technology should be used.

1.1. Agents' Origin

In the beginning of the 80's the Artificial Intelligence scientific community organized a series of meetings, where it was proposed another sub-area of research: the Distributed Artificial Intelligence. This new field resulted from the merging of two distinct areas: the Artificial Intelligence and the Distributed Computing. So far the Artificial Intelligence scientific community concentrated the systems' intelligent capabilities is a single entity, the Intelligent System.

According to Davis, the Distributed Artificial Intelligence's goal is the solving of problems where a single problem solver, a single machine or a single computational entity don't seem appropriate [Davis-1980].

Nilsson described Distributed Artificial Intelligence as being related to the type of problem solving in which computation or inference were logically or physically distributed [Nilsson-1981].

Distributed Artificial Intelligence covers two areas: *Distributed Problem Solving* and *Multi-Agent Systems*.

Distributed Problem Solving is based on the assumption that the problem solving task can be sub-divided in a certain number of modules or nodes, cooperating mutually, sharing knowledge about the problem and the solution being produced.

In the Multi-Agent Systems the focus is on the coordination of the intelligent behaviors shown by a community of agents (autonomous or semi-autonomous) in such a way that they will be able to share knowledge, abilities, goals and plans in order to take actions or solve problems. Individual agents should be able to reason about the coordination processes involved.

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, promoting agents as an adequate technology for deploying systems on this distributed environment.

1.2. The different "visions" on Agents

The term "agent" is adopted by several different communities, apart from Artificial Intelligence. Sometimes, the word is used in a overstretched manner, as a sort of marketing gimmick to sell a perfectly conventional program as being agent-based. There are cases where one is at pain to see those systems as more than a normal program with the ability of migrating between computational resources.

Sometimes the Object-oriented Programming scientific community defends the point of view that agents are not more than Distributed Objects, hinting that technologies like CORBA, DCOM or RMI are enough.

Lets' consider what is said on the subject by Jennings and Wooldridge:

An object encapsulates some state, and has some control over this state in that it can only be accessed or modified via the methods that the object provides. Agents encapsulate state in just the same way. However, we also think of agents as encapsulating behavior, in addition to state. An object does not encapsulate behavior: it has no control over the execution of methods – if an object ***x*** *invokes a method* ***m*** *on an object* ***y****, then* ***y*** *has no control over whether* ***m*** *is executed or not – it just is. In this sense, object y is not autonomous, as it has no control over its own actions. In contrast, we think of an agent as having exactly this kind of control over what actions it performs. Because of this distinction, we do not think of agents as invoking methods (actions) on agents – rather, we tend to think of them requesting actions to be performed. The decision about whether to act upon the request lies with the recipient*.

The agents' characteristics to be most praised depend on the particular community under consideration. For instance, the Distributed Systems community considers Mobility as a vital characteristic, while in Robotics the reactive behavior and the interaction with the physical world are considered essential, and the Artificial Intelligence field stresses characteristics like reasoning, learning and social skills.

Some authors and Agent technology proponents defend that the concept of Agents-oriented Programming or Agent-based Programming will be the programming paradigm of the XXI century and will be as important as the Structured Programming and the Object-oriented Programming were in the past.

1.3. Agent definitions

A search in the dictionary will provide at least one of the following meanings:

1. a person or thing that takes an active role or produces a specified effect;
2. a person or entity who acts on behalf of another;
3. a means or instrument used by an intelligent entity to obtain a result.

The definitions 2 and 3 seem to be interesting, suggesting capabilities of representation (meaning 2) and intelligence (meaning 3).

From the Computer Science point of view Agents can also be defined in several ways:

(MuBot Agent) – The term agent is used to represent two orthogonal concepts. The first is the agent's ability for autonomous execution. The second is the agent's ability to perform domain oriented reasoning.

(AI, a modern approach) - An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [Russell-1995].

(Maes) - Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed [Maes-1995].

(KidSim Agent) - An agent is a persistent software entity, dedicated to a specific purpose. 'Persistent' distinguishes agents from subroutines; agents have their own ideas about how to accomplish tasks, their own agendas. 'Special purpose' distinguishes them from entire multifunction applications [Smith-1994].

(Hayes-Roth) – *Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions [Hayes-Roth 1995].*

(IBM Agent) – Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires.

(Wooldrige-Jennings) – a hardware or (more usually) software-based computer system that enjoys the following properties [Wooldridge-1995]*:*

- *autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;*
- *social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;*
- *reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;*
- *pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative.*

(SodaBot Agent) – Software agents are programs that engage in dialogs [and] negotiate and co-ordinate transfer of information.

(Brustolini) – Autonomous agents are systems capable of autonomous, purposeful action in the real world [Brustolini-1991].

(Franklin & Gasser) – An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda [Franklin-1996].

(Coelho) – In order to survive, the agents are forced to have decision making capabilities, strategic and previsional, to be able to co-ordinate their actions and to face complex tasks in an effective way [Coelho-1994].

(Minsky) – I will call Society of Mind to a scheme in which each mind is composed of very little processes, called Agents. Each mental agent by itself can only do some simple thing that needs no mind or thought at all. Nevertheless, when we put these agents together into societies and in special ways, this will lead to true intelligence [Minsky-1986].

(Newell) – The main attributes of an agent are: to behave in a flexible way according to its environment; to exhibit adaptive behaviour; to operate in real-time; to operate in a rich and complex environment; to perceive an immense quantity of dynamic details; to use vast amounts of knowledge; to contain a motorised system with several degrees of freedom; to use symbols and abstractions; to use natural language; to learn with the environment; to acquire abilities trough development; to live autonomously within an artificial community; to pay attention to what's around it and to itself.

If we take these definitions into account, then became clear some of the characteristics that can be associated to an agent, such as, for instance, the sensorial capability, the ability to act and react upon the environment, the autonomy or the social skills that allow the interaction with other agents.

Some definitions are more focused on the Agent as an individual entity, being closer to the sense of an agent as an entity that represents something or someone (for instance, an agent to perform searches or purchases in the Internet on our behalf). This approach is said to be associated with the Intelligent Agent concept.

Other definitions are more geared to the agents' social skills. In these cases, the focus is not on the agent itself but on the agents' community, and hence the most valued characteristics are the argumentation, negotiation and conflict resolution capabilities. One can say that this approach is centred in Multi-Agent System concept.

2. Agents' classification

Agents can be classified according to a set of possible characteristics. It is hard to find an agent presenting all the proprieties cited in the previous chapter, but it is clear that some of them are key in defining the agent's concept. The Table 1 lists all the main characteristics that can be used to describe an agent.

Properties	***Meaning***
Sensorial capability	Has sensors to gather information about its environment
Reactivity	Feels and acts, reacting to on-going environment changes
Autonomy	Decides and controls its own actions
Pro-activity	Is goal driven, goes beyond reacting to the environment
Persistency	Exists during long periods of time
Social skills	Communicates and co-operates with other agents or even people, competes, negotiates
Learning	Is able to change its behaviour based on prior experience
Mobility	Is able to move from one computer to another
Flexibility	Its tasks don't need to be pre-determined
Agility	Ability to swiftly take advantage of new unforeseen opportunities
Character	Credible personality and emotional behaviour
Intelligence	Ability to reason autonomously, to plan its actions, to correct its mistakes, to react to unexpected situations, to adapt and to learn

Table I – Agents' characteristics/properties

Each of these characteristics will be discussed in the next section .

2.1. Sensorial capabilities

Agents must have sensorial capabilities. When the agent interacts with the physical world, for instance, controlling a robot or a room environment, the sensorial input is supplied by physical sensors (proximity or tact sensors in robot's case, temperature, humidity or smoke in the room's case). Sensors could be of the on/off type or be able do discriminate between different digital levels or even be of the analogue type. Armed with those sensors, the agents can "feel" the environment, be it a structured one (previously known) or not (for instance, a rover robot exploring a planet like Mars).

This concept of sensing abilities can be extended, when no external physical devices are controlled, to a computing system, which will be in this case described as a Software Agent. An agent can monitor the use of an hard-drive or look for the telltale of a virus. We may

consider those abilities as being of the sensorial type, like an agent having a virtual sensor for detecting events in a computational environment.

2.2. Reactive and Deliberative Agents

The Reactive Systems concept come from the Robotics field, where Brooks introduced a multilevel reactive architecture. Previously "intelligent" robots followed the cycle "Feel→Plan→Act", where all actions were planned and, as a consequence, the response times could sometimes be quite high. Brooks proposed the concept of *reactive behaviour*, according to which the system reacts to stimuli without a deep planning.

One can say that the reactive behaviours allow the simulation of human reflexes. We don't need to modify the plan we follow to go back home just because the car ahead suddenly braked. We just have to brake or swerve to avoid it and then retake our normal course. The reaction is immediate and no change of plan is needed.

Brooks architecture is based on the assumption that the triplet Feel/Plan/Act is not an cycle but composed of tasks to be performed in parallel. While we brake not to crash into the next car we can simultaneously start planning a different route to avoid the traffic jam and at the same time visually check whether an alternate street is unobstructed.

Both in physical (mobile robotics) or virtual (virtual reality, "pure" software) environments, that exhibit great dynamism, it is advisable to employ agents with reactive behaviours in order to have them reacting quickly to change. However it can happen that the reactions are not the most wise (for instance, to drop a pan full of boiling water because the handles are too hot or to brake a car when driving on ice). Therefore, the complete inhibition of a deeper reasoning in favour of immediate reactions can induce problems.

The Reactive Systems can come in three flavours:

- Purely Reactive Systems - there is 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 reactive behaviours or add new ones. These systems can exhibit some adaptive and learning capabilities.

The type of planning that takes reaction into consideration is called Reactive Planning, or Tactic Planning. It is usually associated to the concept of Online Planning, that deals with planning during execution.

The opposite of Reactive planning is the planning that is made a priori and does not accept any changes throughout the execution. It is called Strategic or Deliberative Planning and is prepared off-line.

Deliberative Agents therefore can be seen as the opposite of Reactive Agents. They keep an internal representation of the world around them, using an explicit mental state modifiable be symbolic reasoning. A purely deliberative hypothetic agent wouldn't change a previously drawn plan just because the environment had changed. Therefore, in practice, it is hard to find 100% reactive or 100% deliberative agents, most of them being hybrids closer to one or other end of the spectrum.

2.3. Autonomous and Semi-Autonomous Agents

When we say that an Agent acts on behalf of someone, we are implicitly admitting that it has a great degree of autonomy. Autonomy is, in fact, universally accepted as the most important characteristic of an Agent, although being not enough, on its own, to characterize it.

Let's recall Franklin and Gasser definition:

*(Franklin & Gasser) – An autonomous **agent** is a system situated within and a part of an environment that senses that environment and acts on it, in pursuit of its own **agenda** and so as to effect what it senses in the future.*

A fully autonomous Agent is an agent that doesn't need others to assure its existence or persistency. It doesn't follow from here that this agent is capable of doing everything, it just means that it will not freeze just because others (agents or human beings) were not capable of fulfilling a certain assignment.

Autonomy is essential in situations where the online human intervention is difficult or even impossible, like controlling a space robot or a spaceship. Autonomy can also be key in situations where human operators can become "frozen" due to the critical nature of the situation (for instance, accidents in nuclear power plants could have operators taking wrong decisions due to psychological pressure).

Agents may also be semi-autonomous. In this case they will partially depend on others and on human beings. Semi-autonomous agents don't make sense individually because they depend on others to fulfil different functions or to validate agents' decisions.

Figure 1 gives a graphical view of a possible classification of Autonomous Agents.

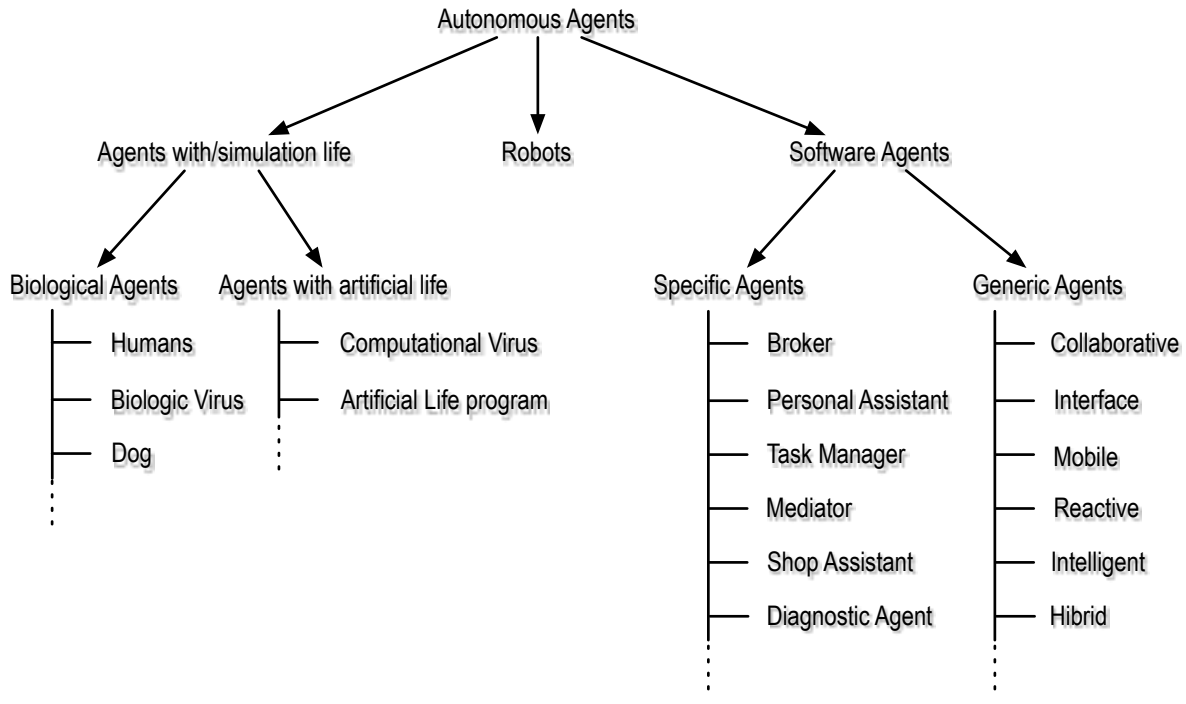

Fig. 1 Autonomous Agents Classification [Franklin-1996]

4. Proactive Agents

A Proactive Agent doesn't limit itself to react to the environment, it has its own vision and goals. It is an intervening agent, capable of modifying willingly the environment where it operates.

A web search agent working on behalf of someone is not proactive because it doesn't change its environment. However, if this agent is able to produce its own pages, or to try and convince other sites managers that the information they offer is incorrect or incomplete, then we can say that we are dealing with a proactive agent.

2.5. Persistent and non-persistent Agents

Persistent Agents are agents that exist permanently.

A computational process is born (process creation) and can go through several intermediate stages throughout its existence (running, on hold, frozen while waiting for something to happen) until being no longer useful and vanish (end of process). It is the case with most of the agents: it has a life cycle relatively short, and is removed when its task has been fulfilled.

A persistent agent has a greater time span. In theory, it should exist permanently. Let's look at some examples:

- In an industrial production system an agent representing an sales order or a production order is not persistent, but an agent representing a machine or production line is.
- An agent responsible for alarm processing in a power system Control Centre is a persistent agent. An agent in charge of planning the power system restoration after a serious incident doesn't need to be persistent.
- An Electronic Commerce agent representing a certain vendor will be persistent. An Electronic Commerce agent representing an occasional buyer can't be considered as persistent.

2.6. Agents with social skills

Usually, an agent doesn't exist in isolation. It is common for it to have to interact with other agents, similar or different. In the same way that we talk about societies of individuals, we can also consider communities of agents. This is the basis for the concepts of Multi-Agent System and Holonic Systems.

Agents need to gather information and knowledge and therefore they need to establish communicate between them. Several technological alternatives for communication are available: peer-to-peer (one to one) communication, broadcast (one for all) communication, blackboard based communication. Sockets, shared memory and other techniques can be used.

Agents must use a common language in order to be able to understand themselves. They must share vocabularies and taxonomies allowing a consistent dialogue (ontologies). They will also need knowledge interchange formats (KIF) and knowledge query languages (KQML - Knowledge Query Manipulation Language) as the basis of agent communication languages (ACL) .

Cooperation is a concept that is defined at a higher level than communication. Agents cooperate in order to try and achieve their common goals or to get some benefit. Obviously, cooperation needs some kind of communication.

A cooperative agent needs to know what its skills are and to have an idea about what tasks can be accomplished by other agents. This information can be stored in agent's data or be obtained by asking a specialized agent about it. Cooperative agents are able to share both tasks and results (data and knowledge).

Agents can compete between themselves (they can represent, for instance, electronic commerce companies selling identical products in the web). In that case, agents must have increased abilities to monitor the environment evolution, namely of being able to watch closely over 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 several restrictions (cost, time, quality, etc), conditions and penalties. Negotiation between agents has been identified since the beginning of the Distributed Artificial Intelligence, as can be seen in the Contract Net Protocol [Davis-1983].

In a community of agents it is common for conflicts to occur. Some examples of conflicts are:

- 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 and Knowledge - agents have different views on the same situation or reality.

Communities of cooperative agents can be divided in tightly coupled systems and loosely coupled systems. In the former case, agents are very dependent on each other and, as a consequence, if one agent fails there is a strong possibility that the multi-agent system also fails. In the later case, agents have a greater autonomy; therefore, if an agent fails the system will be able to find a solution, although of lesser quality.

Sociability, or the ability of an agent to operate with other agents within the same society, is the key to differentiate between an intelligent software system and a system of intelligent agents.

2.7. Agents that learn

The ability to learn is one of the agents' characteristics that Artificial Intelligence community likes to emphasize. In a similar fashion that we are able to keep learning while we interact with our environment and gathering more information and knowledge, also the agents must be able to evolve, i.e., to adapt its behaviour according to prior experience.

Agents' learning abilities shouldn't be limited to the ones typical of Intelligent Systems, where the focus is on learning about a specific domain (for instance, learning how to classify a customer requesting a loan or how to foresee water consumption during a certain period of time). Agents should be able to display higher levels of learning capability like, for instance,

learning to recognize an agent as not reliable in what concerns lead times of previously contracted tasks.

Let's imagine an agent that on a given situation takes a decision that later proves to be the wrong one, becoming aware of that. If, the next time the same situation occurs, the agent takes a similar decision, then this age cannot be considered intelligent, because it was not able to learn.

Some of the approaches used in automatic learning are:

- Case-based learning;
- Observation-based learning;
- Learning using explanation;
- Symbolic classification;
- Neural networks.

2.8. Mobile Agents

Mobility is one of the key agent characteristics from the point of view of Distributed Systems and Computer Networks scientific communities. One should notice that we are dealing here with software agents, so the mobility we are talking about is not of the physical kind (like a mobile robot). Agent Mobility is defined as the ability to transfer itself to a different computational location. Not to be confused also with the concept of software portability.

A mobile agent is supposed to be able to leave the computational environment where it currently operates and move itself to a different one. It is a program that can migrates from one machine to another in a heterogeneous environment. The agent chooses when to migrate and to where. It can suspend execution at an arbitrary point while in a certain machine and transfer itself to another one, reactivating itself upon arrival and restarting at the same point where it left.

When agents move across a network they use resources. Attention should be paid to avoid the overuse or waste of these resources. A realistic mobile agent should be efficient when evolving in a distributed and heterogeneous environment. During its life cycle, an agent may visit machines of different types and operating systems, that are used by organizations with different policies and goals. Mobile agents must be prepared to deal with that kid of situations.

One example of such an agent is the one that helps in the management of IT infrastructure of some organization. It will work on behalf of an IT manager going into the computers present in the network or invoking daemons in all the machines.

An internet search engine is not mobile in the sense that it doesn't abandon the computer where it started, although it will search for information located in a lot of computers. A virus, though, has mobility characteristics because it is capable of spreading into different computers.

Programming languages like Java made more clear the need for Agents' mobility.

Mobile users have raised a new problem not present in software applications geared to fixed users. Mobile users care about the cost and reliability of communications. Mobile Agents technology may be used to address mobile users concerns.

Mobile computing is, as suggested by the name itself, a field that deals with the mobile aspects of computing. Mobility can be classified according to one of these types:

- only users are mobile, computers being static;
- although computers and users being mobile, both are static during the session;
- users are mobile during the session, raising problems like message forwarding and connection management.

Mobile agents platforms have bigger problems in what concerns fault tolerance. Mobility also raises some issues about access priority and security. Mobile agents can be vulnerable to hostile attacks. These threats can materialize in:

- taps for obtaining confidential information (with or without access keys);
- false identification;
- mobile agents clones
- transaction duplication;
- fraud;
- user information misuse.

Some of the first mobile agents systems were AgentTCL [Gray-1995], Concordia [Mitsubishi-1997] e Odyssey [Odyssey].

2.9. Flexible and Agile Agents

Flexibility is the characteristic that enables agents to easily switch tasks. In a Industrial Production System, for instance, when a new order arrives for a known product, a higher priority can be given to the correspondent production order, even if that implies that the already issued production orders have to be rescheduled. An agent that is able to react this way would be considered as flexible.

Agility is a somewhat different characteristic and it has to do with the ability of an agent to satisfy quickly a new, previously unheard, request or to grab a new opportunity, even if it means that a new task, not foreseen in advance, has to be performed.

A typical example of agility is the concept of Virtual Enterprise where several companies get together to address a new market niche where a differentiated product is needed.

2.10. Agents with Character or Personality

Agents are often used to perform certain tasks on behalf of human beings. Will it then be admissible to wonder if one can assign to an agent a specific character, a personality or an emotive behaviour? If we create an agent for shopping in an Electronic Commerce environment we can give them small particles of character. The agent can, for instance:

- avoid buying products coming from certain countries;

- avoid buying products made by companies that finance certain political or sport associations;
- avoid closing deals with companies whose advertising practices have some kind of characteristics;
- avoid buying from non environmental-friendly companies.

When agents are bound to interact with human beings a great importance must be given to the interface design. Some work has been done in the development of anthropomorphic computational personalities that seem to exhibit some kind of emotional behaviour, for instance, changing facial expression when the user refers something unpleasant. Strides have also been made towards the user's emotional status identification. Agents with the ability to exhibit and recognize emotional behaviour need to access technologies like voice recognition, natural language, computer vision and graphical computation.

The introduction of anthropomorphic agents poses several 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?

Interface agents with anthropomorphic elements are still rare, being more common in research projects. Some researchers have the opinion that technologies like voice recognition, natural language understanding or automatic learning haven't yet reached the degree of maturity needed to assure the interface agents' success. Moreover, some critics defend that anthropomorphic agents may generate confusion among developers and users, create anxiety in users, decrease his control capabilities, contribute to a lesser sense of responsibleness and reduce user's commitment.

This is nevertheless a very active development area. Some proponents defend that anthropomorphic personalities can be useful. Walker, for instance, studied the way users answered questionaries online, both of the conventional type and the ones using anthropomorphic interfaces [Walker-1994]. The conclusion was that the anthropomorphic interfaces lead users to pay more attention to the questionaries, to commit less errors and give more suggestions.

There has been lately a large debate about Emotionality versus Intelligence. A person with a higher IQ or greater skills can be less stable emotionally than another person less intelligent or skillful. In critical situations, human emotional behaviour is key. Let's consider, for instance, the reactions of the pilot of an airplane in trouble or the operator controlling a nuclear power plant in a complex situation. Software is not always able to sense those situations and that can be good or bad. Care must be taken when transferring emotional behaviours to agents.

2.11. Intelligent Agents

Agent intelligence is a characteristic that can be defined using other characteristics. Let's illustrate this with the following definitions from Hayes-Roth and IBM:

(Hayes-Roth) – An intelligent agent performs continuously three functions: perception of the environment's dynamic conditions: actions that affect environment's conditions; reasoning abilities to interpret perception results, to solve problems and determine which actions to execute.

(IBM Agent) – Intelligent agents are software entities that perform a set of operations on behalf of an user or program, with some degree of independence or autonomy, using some knowledge and representation of user's wishes or goals.

This almost reminds us of the Turing Test used to try and verify wether or not a given system was human. Trying to adapt the terminology to the current reality, one could say that when we are not capable to determine wether or not the output of an agent comes from a human being, then we will be facing an intelligent agent.

IBM's definition includes the term *knowledge*, very important since several ago in the field of Knowledge Based Systems (Expert Systems, for instance) or in Knowledge Management nowadays. In fact, the more knowledge we are able to impart to an agent the greater the possibility of reaching something closer to a truly intelligent agent.

A clear distinction between Agents and Intelligent Systems is that it is not enough for the intelligent agent to display intelligence in its specific domain. The intelligent agent also needs to know how to interact within an agent community or with humans, to know which agents or persons are trustworthy, how to negotiate with agents or entities from different backgrounds, how to deal with conflicts, how to share knowledge, and so on.

In Figure 2 Cooperation, Learning and Autonomy are presented as the main characteristics of an Intelligent Agent.

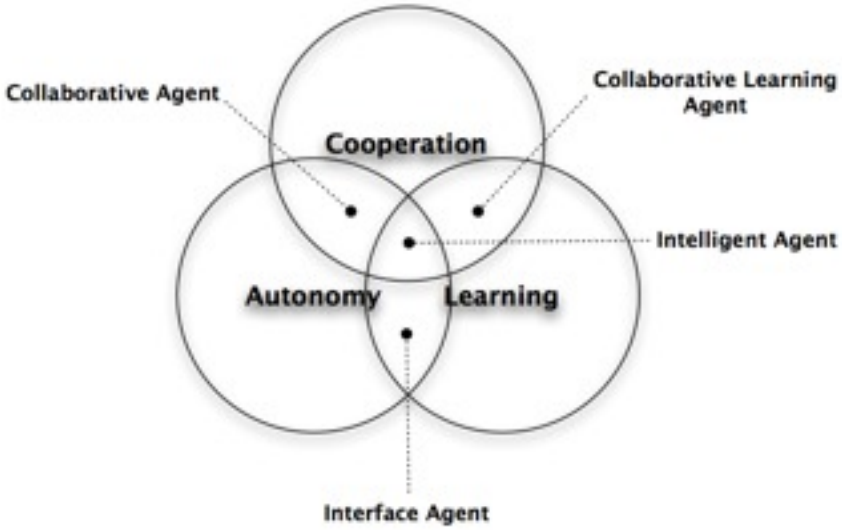

Fig. 2 - A possible vision on Intelligent Agents [Nwana-1996]

Several research efforts has been focusing on the analysis of human and agent intelligence. How do we evaluate another person's or agent's intelligence? If we pose two persons or agents a set of questions, we will tend to consider as more intelligent the one that gives more correct answers and does that quicker. Likewise, we will more easily trust the answers given by a human being or an agent that is branded as an expert. The intensity of the criticism that a certain solution faces can also skew our evaluation. There is often the tendency to consider that the more critical analysts are the ones more competent. The same considerations we can make about the subjectivity on judging other people's intelligence can also be made when we are dealing with agents.

The very same fears that existed in the past about robots and Artificial Intelligence seem to have been replicated now with the Agents. When we consider intelligent agents capable of having opinions or deciding about purchases or sales of products or stocks, it becomes clear that the automation of human tasks by computational means is evolving from the more repetitive and tedious to the value-added, knowledge-intensive kind of tasks.

3. Agents and Multi-Agent Systems' examples

This chapter will contain some brief examples of Agents and Multi-Agent Systems. The reader will be able to better identify the concepts and characteristics previously described. Possible applications of these technologies to Electronic Commerce, Manufacturing Systems and Traffic Control will be discussed.

3.1. Electronic Commerce

Agents are software entities to whom were given autonomy and intelligence enough to be able to perform specific tasks with little or no human supervision. An agent usually interacts with a continuously changing environment, while representing the interests of some entity.

The convergence of Computing and Telecommunications technologies, specially after the internet generalization, had a strong impact in the way commercial transactions are conducted. Internet growth surpassed the most optimistic expectations. From 17 million in 1992, the number of internet users reached 1262 million in November 2007, of which 343 million in Europe only[1]. Most of internet users already tried to buy something on the web. As early as 1995, 5.2 million european consumers used the internet to buy products or services with a total value of 3032 million €. The equivalent figures for 2002 were around 28.8 million users spending an estimated value of 57210 million €. These figures reflect only direct purchases by internet users and do not include the business to business (B2B) transactions, far more developed that the private ones. Although totally reliable data is not available, the total Electronic Commerce figures for 1999 may have totaled 95 thousand million USD and in 2003 this figure may have been multiplied by a factor of 100 to 200.

Several tools for online business setup have been developed, like Merchant from Microsoft, Net Commerce from IBM and Dynamo from ATG.

I makes sense to use agents in Electronic Commerce because they can automate several tasks, like visiting web sites or gathering data from the internet, much more quickly than human beings. This is particularly important in markets where the prices keep changing all the time (stock markets, raw materials exchanges, etc).

Security issues with agent-based Electronic Commerce should be seen under a different light than with conventional applications. The latter are focused on transaction security (SET or SSL standards). Agent-based Electronic Commerce applications must also offer this kind of protections but when we see agents as representing people or organizations, we must consider eventual "damages" that agents may provoke. Agents may overspend, may buy unwanted or illegal products or may close deals with unreliable vendors. Therefore, security in agent-based Electronic Commerce must be addressed differently. The main concern is not at the transaction level, although this must be assured by other means, but at the trust level: will we trust an agent with our credit card data?

---

[1] Source: http://www.internetworldstats.com/stats.htm

When we talk about Electronic Commerce we are dealing with two different levels: individual purchases and transactions between companies (B2B).

With individual purchases the main difficulty is how to personalize the agent. The normal procedure is for a customer, using a internet-enabled computer, to make a few searches using browsers, to identify and compare products and ultimately making choices. We will have to distinguish between standard products (a book or a CD) and differentiated products like clothes. In the first case, agents' introduction will much easier. A script for the agent to perform could be:

Search all the websites selling the product X
Identify the one with a lower price (including freight costs)
Make the order if the price is lower than PrX.

Obviously it may be more complicated than that if the user wants to consider aspects like the delivery time, the trust that the supplier inspires, the payment terms or the existence of eventual bonus. The problem may get even more complex if the user wants to order a set of different products that may be not all available from all the suppliers. In this case, the agent must show a greater level of intelligence in order to be able to select the right suppliers.

When we consider the Electronic Commerce dealing with individual purchases of differentiated products, the use of agents for automatic decision making becomes much more complex. The main difficulty is that no consumer is willing to buy an unknown product. It is hard to imagine someone giving an order to buy "a pair of brown shoes size 38". In this case agents must be seen solely as entities looking for products that fit into a certain description given by the user.

In any case, so far we are considering agents as individual entities even if they may actually be dealing with virtual instead of human vendors.

We can also center Electronic Commerce on the vendors side. Here the agents should be responsible for the product advertisement and transaction security. A more complex agent may even be able to assure more demanding tasks like transaction analysis or competition monitoring. It may, for instance, reach the conclusion that the sales of a certain product decreased because of an alternative offering by a competitor at a lower price. This way, quicker reactions to market changes will be possible, leading to lower losses or bigger gains.

Seller agents can also model the customer, identifying a profile. For instance, if it is known that whoever buys products P1 and P2 falls usually within a certain profile of people characterized by also buying the product P3, then if someone buys P1 and P2 it may be assumed that person will be also interested in P3. Data Mining techniques are adequate to extract conclusions of this type, giving a better insight into the market and consumers profile, enabling "targeted" marketing. It should be noted that if a real, physical store is converted into a virtual one the vendor looses direct contact with the customers and the correspondent possibility of getting inputs from them. Data Mining can be used to automatically recover some of this visibility, analyzing past transactions history.

Buyer agents must know how to find out who sells the required products and how to place an order. This can be accomplished by storing in the agent a list of websites and information

about the ordering procedures. If it is required for the agent to search for new vendor sites they must be able to use a language for automatic content reading (like, for instance, XML).

In the Electronic Commerce between companies the products have well defined characteristics. In the auto industry, for instance, the car producer buys components (like tyres and cable assemblies) and raw materials (like glue and metal sheets) from suppliers. The supply chain can include, for instance, electric cable manufacturers that need copper and PVC in order to supply the cable assembling companies. Then, if agent technology is being used, we can start talking about a Multi-Agent System.

In this environment, buyer agents must be much more careful with potential vendors than in the individual type of electronic commerce. Parameters like component quality levels and delivery time reliability are vital. If the vendor selected to supply cable assemblies delays the deliveries, then the automaker, unless it has an adequate security stock, will be forced to delay theirs. Therefore, electronic commerce between companies requires a great care in selecting the right supplying partners. Usually, agents already know with whom negotiation must be established but they may face situations where they have several suppliers for the same type of products, and they will try and decrease prices and increase the assurance of deliveries being met.

Agents need to embody enough intelligence in order to close good deals. They may also be given argumentation capabilities in order, for instance, to be able to schedule deliveries in a way that decreases inventory costs. These agents may need to monitor security stock levels of components and raw materials.

In an Electronic Commerce environment can also exist intermediary agents who try to make easier the establishment of deals between buyers and sellers. Buyer agents and seller agents can have the ability to negotiate prices, delivery times, payment terms and so on. Negotiation may or may not involve intermediary agents but in the last case, they must be trusted by both sides (buyers and sellers).

3.2. Robotics

We will demonstrate the use of agents in Robotics with a few examples:

The first example is a robot that identifies individual components by means of a computer vision system and sensors and that is able to perform manipulation and assembly tasks. We can have agents with different capabilities, namely the component identifier, the trajectory planner (used to avoid collisions), the assembly planner and the execution controller. There can be more than one robot performing tasks, which makes the need for cooperation more pressing (for instance, in order to avoid a collision between two robots). Additionally, two robots may also cooperate to together perform a task that couldn’t be done by a single robot.

The second example is about a robot community that is exploring a uncharted geographic area (like a new planet ground or the sea floor). Each robot must have autonomy and intelligence in order to decide where to go or how to avoid collisions with obstacles detected by its sensors. In that sense, these robots can be seen as agents. These agents can communicate directly between themselves, sharing data and knowledge. A robot can, for instance, inform the others about the existence of an obstacle with certain dimensions in a certain position. It

should be noted that these agents, unlike the ones from the previous examples, are functionally identical. In this case, the possibility of conflicts arising can be greater. Another robot can, for instance, answer saying that had already passed by the coordinates indicated and found no obstacle.

The last example is about robotic football (RobotSoccer). We can imagine a team of robots, each one associated with an agent, with sensors and actuators. A camera grabs an image from above that is fed to another agent (the coach) who will keep sending instructions to the several agents, in order to correct some aspects previously planned. Although the agents being identical, they may have to perform different functions (goalkeeper, defender, forward). These agents can show reactive behaviors (for instance, a goalkeeper must react swiftly when a forward from the opponent team comes with the ball from the other side). Here, we can see the differences between strategic and tactic planning. Another aspect to consider is that not all of these agents can be seen as cooperative, the other team's agents are competitors.

3.3. Manufacturing Systems

Manufacturing Systems are characterized by a high degree of complexity, being naturally distributed both from the physical point of view (several machines, assembly lines, factories...) and logical (different products, sales and production orders to be dealt with simultaneously). In this environment it is common to find Multi-Agent Systems and a particular emphasis on the negotiation aspects.

Cooperation and negotiation can be seen at different levels, like, for instance:

- Between resources belonging to the same assembly line or manufacturing cell;
- Between several assembly lines or manufacturing cells;
- Between one company and its suppliers or customers (the extended company);
- Between several companies in order to take advantage of a new business opportunity (agility, virtual enterprise).

Let us consider a simple example of task scheduling in a manufacturing system composed of several resources (robots, CNC machines, automatic warehouses, etc). We can have agents representing system's physical resources and other agents representing the tasks to be performed (be it at sales order or production order level).

A task simulation agent will need some parts stored at the warehouse, must use robots to fetch them and will also require the work of CNC machines. It will be obliged to negotiate with all these resources in order to reach agreements allowing the task to be performed successfully, assuring that requirements like delivery dates, quality levels and acceptable costs are met. The problem grows more complex when several tasks must be negotiated with several resources at the same time. We will have agents representing each task, almost like customer representatives within the company, and also an agent for each resource who is able to negotiate with the different task agents. It is an approach clearly distinct from the centralized intelligent system that may use heuristics of task dispatch but that doesn't represent adequately the intervenient entities. With agents, the Manufacturing System can be seen as a Social System.

3.4. Traffic Control

Air traffic control is a critical and complex application also characterized by a distributed nature. We can imagine agents representing flights, airplane queues or runways. We can also have agents that are able to consider aspects like, for instance, atmospheric conditions. The problem in analysis is clearly of the scheduling type and we can even make a comparison with the manufacturing system:

Flight ⇔ Production Order, Sales Order

Runway ⇔ Machine

Time between liftoffs ⇔ Setup time

Another example of a traffic control application is the urban traffic control system. Agents can be associated to the semaphores, with sensors measuring the traffic flow or the queues size. Agents can negotiate in order to maximize traffic flux. That way, if in a certain crossing one direction there is a much greater intensity flux than in other, then the semaphores' timing can be adjusted. Several semaphores lined up in a row can also coordinate among themselves in order that the cars are not forced to consecutively stop in most of them.

4. Shortcomings of the Agents-based solutions

Some of disadvantages of Agents based solutions are:

- *There is no system controller - agent based solutions may not be adequate for problems where global restrictions must be enforced. when a realtime behavior be assured or when deadlocks must be avoided.*
- *There is no global system perspective - an agent's action is determined only by its local status. Complete global knowledge is not achievable and therefore agents globally cannot reach more than sub-optimal decisions.*
- *Trust and delegation - for someone to delegate tasks on agents one has to trust those agents. Organizations need more experience about using autonomous software agents. The trust building process is a slow one. A careful personification of the agents could be a way of increasing user's trust on agents. Another aspect to be considered is how to assure that agent really is representing us?*

The first two limitations detailed above simply derive from the fact of agents communities being distributed systems. Nevertheless, it is the third shortcoming that limits the most the acceptance of agent based solutions. Let's suppose, for instance, that someone grants us access to a configurable agent specialized in internet purchasing in the user's behalf. Would we be able to trust this agent enough to supply our credit card data and grant it authorization to place orders?

This seems to lead to a new way of developing agents. If the agent's purpose is to substitute the user then the agent should be seen from the start as a Decision Support System that doesn't limit the user's choice but instead makes suggestions, accepts changes and is able to learn from the interaction with the user. This way, the agent will be seen as an adaptive entity with a behavior increasingly close to the user's. The user's trust will increase bit by bit until the agent is deemed fully trustable. This approach, called "Scalable Intelligence", describes the process of converting a simple Decision-Support System in a full Agent.

In a Decision-Support System with Scalable Intelligence the decision maker can use as much system intelligence as he wants. It is like having a scale varying in the range [0,1]. At the lower limit, the Decision-Support System is just a simple tool, without decision capability. With time, the decision maker will try some of the more clever functionalities of the system, being able to limit them as he sees fit. As the decision maker verifies the suitability of those functionalities, he is building his trust on the system and will be able to increase the system abilities, which corresponds to increasing the system intelligence level within the scale. With time more competences will be being added to the system, specially if it is easily adaptable to the user or presents more intelligent characteristics like the ability to engage in an intelligent dialogue and to learn with the decision maker. In doing so, it is increasing the intelligence level until reaching the upper scale level (an hypothetical limit). At that moment, the decision maker fully trusts the system, by that meaning that he trusts an Intelligent System that is able to adequately substitute him in a set of tasks for which the Decision-Support System was originally designed. Therefore, this system is able to be substituted by an Intelligent Agent provided that it is given autonomy [Ramos-2001].

5. Problems with the development of Agents and Multi-Agent Systems

In spite of all the effort made in developing Agents technology, little has been done concerning how to implement those systems (Agents' "Engineering").

Michael Wooldridge and Nicholas Jennings have identified several categories of problems hindering the development of agents oriented systems, among which the following [Wooldridge-1998]:

1) "Political";
2) Management;
3) Conceptual;
4) Analysis an Project;
5) Micro-level (Agent);
6) Macro-level (Society);
7) Implementation.

5.1. Political problems

5.1.1. Optimism with Agents' technology (to "sell" the concept)

Agents technology offer a natural and powerful way of conceptualizing, designing and implementing systems. But they are not a "magical" paradigm. Tasks that are beyond automation and for which have already been tried conventional, not agent-based, techniques will not be treatable only because agents are being used. Agents must be able to solve the specific tasks they were built to address but the technologies available have still the same shortcomings like, for instance, combinatorial explosion or lack of information.

Agents are not able to present reasoning abilities similar to human beings. Therefore, as happened with Artificial Intelligence, agents' technology acceptance will suffer if an exaggerated optimism is displayed.

5.1.2. Agents as a dogma

Agents can be applied to a large array of problems and applications but they are not an universal solution. There are many problems and applications that are adequately dealt with the conventional paradigms (like, for instance, the OOP). There is, in fact a risk of considering that Agents are the correct solution for all kinds of software problems.

The way agents are defined is not the same for everybody and different people deems different characteristics as more important. Who stresses a certain specific characteristic tends to develop applications using that characteristic even when it is not advisable to do it (for instance, mobility may be useless or even not adequate in many situations).

5.2. Management problems

5.2.1. One doesn't know why agents are needed

Optimistic forecasts tend to induce some managers to adhere too easily to the concept, specially when it seems intuitive (as it is the case with agents) and easy to sell (agents that are able to free us from a lot of tasks). Some projects are started without having clear purposes and therefore turn to be very difficult to manage.

This attitude is associated with a lack of clarity of how agents should be used in a way that adds value to the products. It is therefore needed that we lay out clearly the reasons why a certain project should be developed using agents, trying to ascertain what the project can gain from the use of this technology.

5.2.2. One doesn't really know what the developed Agents are good for

One can fall into the trap of developing some new technology or platform using agents before clarifying the exact application that those agents or technologies would have. In that case, the work already done may be totally inadequate (because, for instance, it doesn't fit with the other components of the system) or only a small subset of the functionalities is actually needed (making the solution unnecessarily expensive and complex).

We should try and understand where agents technology is more appropriate and avoid using it on everything situation.

5.2.3. Attempting to develop generic solutions to specific problems

Too often, Agent technology advocates defend the use of generic architectures that may be applied regardless of the specific domain. This is a reminiscence of the early times of Artificial Intelligence when many researchers thought that it would be practical to develop generic problem solvers.

5.2.4. Confusion between prototypes and systems

Once an application has been found for which agents technology seems adequate, it is a reasonable simple task to develop a prototype (a few agents performing some useful and simple tasks). Nevertheless, the transition from prototype to system is much more complex and the rapid prototyping that was done may prove misleading. Agents deal with several complex aspects like:

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

5.3. Conceptual Problems

5.3.1. To believe that agents are a miraculous solution

There is sometimes the idea that a certain technology will promote a great development (an order of magnitude greater than the previous ones). This is starting to be the case with agents technology. In spite of being very promising and even considered by many as the next

programming paradigm, this technology still needs to be thoroughly tested and yield consistent results.

5.3.2. Confusion between commonplaces and concepts

One of the positive aspects of agents technology has to do with their being a intuitive concept. This can lead people to think that they fully understand it when in fact they don't. The use of clichés, very usual in management, has sometimes the effect on the designing of new systems, in an attempt to make them appear as being state of the art.

5.3.3. To forget that we are developing software

Being agents technology still a research area, there are still no adequate techniques to assist systems development. Agents' projects tend to involve tasks like finding the adequate architecture, developing cooperation protocols or improving the coordination and coherency of the multi-agent activity.

Usual Software Engineering processes like requirement analysis, specification, project, verification and test are often forgotten, with dire consequences. It is true that those processes were not conceived with a community of agents in mind but it is better to use what is available than nothing at all. To disregard Software Engineering can lead to system failure not because of the agents per se but because good practices of software development were not followed.

5.3.4. To forget that we are developing distributed software

Distributed Systems are known to be complex systems, difficult to project and implement. Multi-Agent Systems are inherently distributed. The problems plaguing Distributed Systems don't simply vanish just because we are using agents. On the contrary, multi-agent systems are even more complex than distributed systems. The one that projects and develops multi-agent systems should be proficient in the core techniques of distributed systems (synchronization, mutual exclusion, resource sharing, deadlocks, and so on).

5.4. Analysis and Project problems

5.4.1. Not to explore associated technologies

In the development of a Multi-Agent System a good part of the effort should be devoted to the use of different technologies that may enhance the multi-agent system and need to work well in order to improve system performance.

5.4.2. The project doesn't explore the concurrency

It is common to find multi-agent systems where an agent does some processing, produces some results which are passed to another agent and falls in a sleeping mode. The agent receiving the results processes them and sends the results to another agent, entering after that in sleep mode as well. The process repeats itself until the desired outcome is reached. These systems don't use concurrency and could be designed as a conventional centralized program, with several modules.

Concurrency is precisely one of the main advantages of multi-agent systems. It gives the ability of dealing simultaneously with several perspectives and purposes, to answer and react

to the environment at different levels and to consider complementary methods of problem solving in a coordinated mode.

5. Micro level problems (Agent)

5.1. We want to have our own agents architecture

There may be the temptation to consider that none of the existing architectures answer the requirements of our problem and that it will be necessary to specify and project a new one. The "not invented here" syndrome leads to the unnecessary development of new architectures because one only trusts in-house developments. However, the development of a new architecture can take years to complete and there is no assurance that such an effort will be rewarded and its results reused.

It is advisable to take the time to study existing architectures and to acquire the respective licenses or to implement already tested architectures.

5.2. To presume that the architecture being used is a generic one

The people developing an architecture may have the temptation to believe that it may be a generic one. And it is all the more tempting if the architecture was successful in a certain domain. However, this architecture's characteristics may be inadequate for a different domain. If we developed an architecture successfully used in a specific application we should try to understand the reasons behind that success in that particular domain.

5.3. Agents use too much Artificial Intelligence

The reading of articles and reports with examples of Artificial Intelligence techniques in Agents may lead to the conclusion that the more Artificial Intelligence is embedded in our agents the better will be the performance of that community of agents. Many of those Artificial Intelligence techniques are not robust enough and that can impact negatively the success of our project. It is wise to follow the saying "think big but start small". It is better to try first a solution with little Artificial Intelligence and slowly embed more with time.

5.4. Agents have no intelligence

Too often, for reasons of marketing, solutions that would otherwise have been classified as mere distributed systems are now described as multi-agent systems. It is also common to see web pages with some additional processing classified as agents. This is incorrect because voids the term "agent" of all the meaning. Besides, that attitude deceives the users that bought a conventional software wrongly classified as agents.

5.6. Macro-level problems (Society)

5.6.1. We see agents everywhere

Once we grasp the concept, we may tend to believe that almost every system may be seen as an agent. In the limit, we could have an agent to make sums and another to make subtractions. This line of thought will end creating so many agents and consuming so many communication resources that the resulting systems will be very inefficient, if not utterly impossible to build.

Generally speaking, we should have a coarse granularity for the agents due to the fact that an agent should embody coherent abilities and functionalities. The idea of decomposing agents in other smaller agents may be interesting but may lead to computational overloads that end up giving disappointing results.

5.6.2. We have too many agents

One of the most interesting aspects of the community of agents has to do with the occurrence of emergent behaviors when we have a great number of agents. A behavior is said to be emergent when it was no globally foreseen but ends up occurring as a result of the agents' processing and interaction.

These behaviors may be a positive thing, generating useful work, but they may also create chaotic situations. It is difficult to control the dynamics of a system when we have a big number of agents. Besides, we will also have the communications overload problem, already discussed.

5.6.3. We have too few agents

Too often the developers of systems using agents fail to understand the gains we can get from the concurrency factor and tend to develop communities with too few agents, each one concentrating too many tasks and functionalities.

5.6.4. We spend too much time creating the infrastructure

Agents' technology an emerging area, only now start appearing the first platforms with enough credibility for the development of agent-based solutions. However, in the past these solution were not available and every Multi-Agent System project had to include a significant workload and budget just for the development of the basic platform (messages, runtime control, monitoring, etc).

Even now, the existing platforms may be risky bets because they may well start and be discontinued sometime after. Some of them are very good for monitoring what is happening within the multi-agent system but are, at the same time, very heavy, while others, on the other hand, are lighter but not very useful for the developer.

It is also common to find communications infrastructures developed by people from the Artificial Intelligence area, instead of developers coming from the distributed systems and communications, what may hinder the stability of the infrastructure.

5.6.5. The system is chaotic

Sometimes one thinks that by simply adding a set of agents without interaction, a community of agents has been created. However, most of the multi-agent systems need some kind of structure, definition of competences and control, in such a way that the agents community will be able to do some useful work, reaching its purpose.

5.6.6. Confusion between simulated and real parallelism

The multi-agent system project often starts in a single computer, using independent processes or threads. this may be easier in a initial phase of the development because we don't need several networked computers. However, going from the parallelism simulated in a single

computer to the real parallelism in a network environment supposes an order of magnitude increase in the complexity of the multi-agent system control mechanism.

7. Implementation Problems

7.1. Tabula rasa

When we start the implementation phase we often make the mistake of admitting that everything must be developed from scratch. However, this is not the case in many situations, where critical components exist already, have been thoroughly tested and proven reliable. Those components, although eventually obsolete, are difficult and costly to substitute. It should then be studied how to use these “legacy” systems without, in most cases, having the possibility of interfering with them.

7.2. Ignore the standards

The standardization in the agents field is in a primitive state. There are proposals but the standards have not been globally accepted what prevents them from being used. This is a major problem when we need to have agents working together, that have been developed by developed by different groups.

In spite of the lack of standardization, it is advisable to try and follow some of the proposals related to agents interaction (KIF, KQML, ACL).

6. Agents and Multi-Agent Systems' architectures

An agent architecture determines its internal structure, by defining the modules that handle the various tasks to be performed by the agent and the way these modules interact in order to intervene on the agent's environment.

Usually, an agent is involved with one or more communities. Therefore, it makes sense to also address the issue of multi-agent system architectures, which define the way the agents are organized in order to collectively solve a problem.

We may establish an analogy saying that the agent relates to the psychology of an individual whereas the multi-agent system relates to the sociologic aspects of a group of individuals.

6.1. Agent's Architecture

In this section we are going to study some basic agent architectures, not in a thorough way but trying to describe their main modules.

The first one to be studied is the **BDI** architecture (Belief, Desire, Intention) illustrated in Figure 3. The basic concept behind this architecture, that should be considered as a deliberative one, is the description of an agent's internal processing state using *mental categories*. This way, it is possible to establish a control mechanism by which an agent rationally selects the actions to be performed according to its representations. The mental categories to be considered are the *beliefs*, the *wishes* and the *intentions*. Additionally, higher level categories may be defined, like *goals* and *plans*. Let's describe those categories in more detail:

- *Beliefs* - agent's expectations about the current state of its environment, taking into account that a certain action may not cause the expected outcome;
- *Wishes* - correspond to an abstract notion that indicates preferences about the future states of the agent's environment or the course of action to be followed. Agents may have wishes that are inconsistent with the reality and obviously don't have to believe that their wishes will necessarily turn into reality.
- *Goals* - it is a more restrict definition than the one for wishes because the agent must consider its goals as plausible, as being achievable.
- *Intentions* - due to the fact that an agent has limited resources, it may happen that not all the goals are achievable at the same time. Therefore, it may be needed a compromise about what priority to assign to each of the existing goals, defining commitments. That process is called intention formation.
- *Plans* - are pragmatic implementations of the intentions. Intentions can be seen as partial action plans. The agent makes a series of commitments stating that it has the intention of fulfilling the specific goals.

Two of the projects using BDI architecture are IRMA [Bratman-1987] and [Rao-1991].

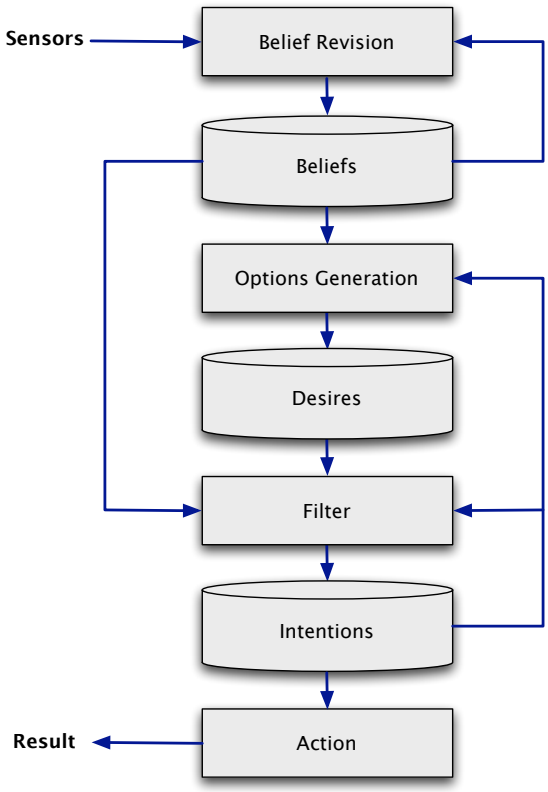

Fig. 3 - BDI architecture

The second architecture to review is the reactive one proposed by Brooks [Brooks-1986] and also known as "Subsumption Architecture". In the 80's, Brooks criticized the trend in intelligent systems design to consider a set of functionalities (like, for instance, sensors, planner, learning module, truth maintenance, execution module) working in series. Brooks proposal is based on a activity-oriented decomposition , with activity producers working in parallel, directly connected to the outside world through sensors and actuators (Figure 4).

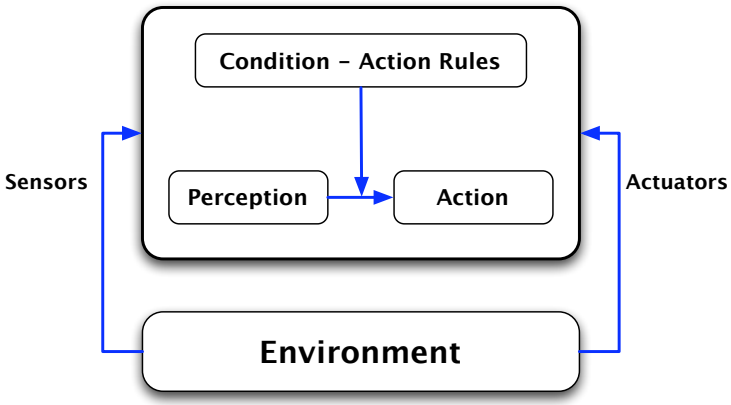

Fig. 4 - Reactive architecture

The third architecture to be described is the ARCHON (Architecture for Heterogeneous On-Line Systems). This architecture was developed in the ESPRIT-II european project [Wittig-1992] and is illustrated in the Figure 5.

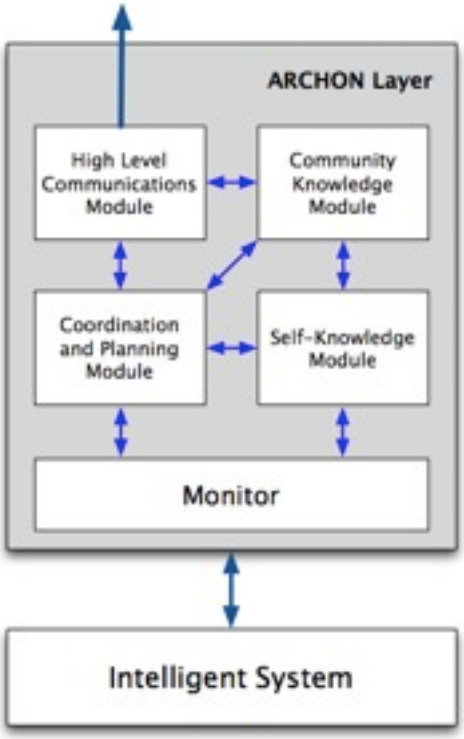

Figure 5 – ARCHON architecture

This architecture's basic principle is that any pre-existent system (depicted as Intelligent System in the diagram) may be encapsulated by an ARCHON layer so that he may turn into an agent. This way, an agent will always be made of at least two main components: the Intelligent System and the ARCHON layer. Nothing obliges those two components to run in the same computational resource. The Intelligent System will be responsible for the useful work to be done by the agent (for instance, to recognize an object using a computer based vision system or to assure the task of scheduling air company crews) while the ARCHON layer is responsible for the cooperation with the other agents in the community and to control the Intelligent System.

The ARCHON layer's modules are the following:

- Self-knowledge (SK) module
- Community knowledge (CK)module
- Monitor
- Planning and Coordination module
- High-level communication module

The purpose of the SK module is to hold the representation of what the Agent knows about itself and the tasks it is able to perform. The CK module reflects what the Agent knows about the other agents belonging to the community. Both contain static knowledge (the SK module contains information about what the agent knows how to do, which is usually permanent, and the CK module has the information about who may provide a certain service) and dynamic knowledge (the SK module may contain information about what the Intelligent System is doing now, with the CK module containing information about the tasks being performed by the other agents).

The Monitor is responsible for the interaction with the Intelligent System (IS) associated with the Agent and the control of its activities. It receives requests and data coming from the Planning and Coordination module, schedules the tasks to be performed by the IS (and has the power to interfere with the way the tasks are accomplished), receives the results from the IS and transfers these results to the Planning and Coordination module.

The Planning and Coordination module decides when and how the Agent must establish a cooperative relationship with the other community agents. It is the module responsible for the global assessment and the dynamic planning of the agent's activities. It decides to whom the requests must be addressed and what restrictions must be associated with those requests.

The High-level communication module defines how the dialogue between the Agent and the other members of the agent community will be done. It uses a message passing mechanism (socket based) with three additional services: intelligent addressing, filtering and message scheduling.

More details on the ARCHON architecture may be found on [Wittig-1992] and [Ramos-1993].

Finally, we will describe the architecture proposed by [Sousa-2000] which is based on the *Holon* concept [Koestler-1967].

The Holon concept was introduced by the hungarian philosopher Arthur Koestler in 1967. The term holon results from the combination of the greek word *holos*, meaning "whole", with the english suffix on that suggest "part", as in proton or neutron. Therefore, holon refers to the whole and the part, betraying a recursive nature: a holon may be made of holons and be part of one. Additionally, a holon may belong to several holons simultaneously. Although having been devised by a philosopher, the holon concept has been adopted by the Intelligent Production Systems scientific community as a model capable of describing adequately a Production System [Ramos-1996]. It is possible to detect a strong similarity between holon and agent concepts. In the next section, we will refer again the subject when introducing the holarchy concept.

In the architecture proposed in [Sousa-2000], described in Figure 6, we can see that he holon/ agent is made of Sensors, Actuators, Protocols, Actions, Reasoning and Knowledge Base. This holon/agent is capable to interact with humans, the environment and other holons.

The Sensors and Actuators blocks represent the system interface, enabling the interaction with humans, the environment and other holons. The block Protocols handles the representation of the information gathered by the sensors (perception). The protocols may be identified by a finite state machine of a communications protocol or a man/machine interaction. This block

allows for the direct execution of actions and the knowledge processing by the Reasoning block.

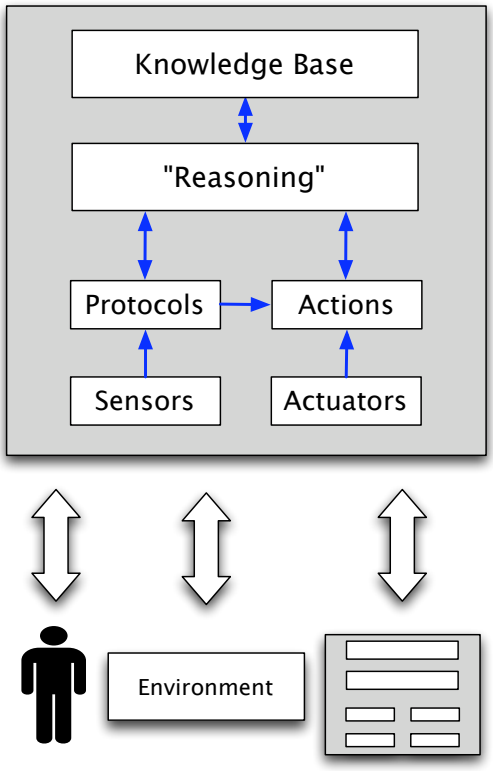

Figure 6 – Holon/Agent architecture [Sousa-2000]

The Reasoning block produces results using its knowledge (Knowledge Base) and the data coming from the sensors. It defines the very nature of the holon, specifying how the holon should behave according to its mental state and the purposes that it has been assigned. The holon's knowledge may have several sources: it may be inherent to the holon's conception; it may be learned from experience or observation; it may come from other holon. The Knowledge Base of each holon must contain a set of general and specific axioms.

1. Multi-Agent System architectures

A Multi-Agent System architecture describes the relationships between agents looking for a solution for a given problem. The architectures may be generic or oriented to a specific problem, they may be more centralized (like a hierarchic structure with a centralizer agent) or more distributed (like, for instance, an "anarchic" structure). The involved agents can be functionally different (complementary) or identical (concurrent, with a greater probability of conflicts). The architecture may be fixe or changeable.

In this section we will look into some Multi-Agent System architectures.

The first architecture to be studied is the CIARC (Cooperative Intelligent Assembly Robotics Community) [Ramos-1993]. This multi-agent architecture was used in an assembly and manipulation system using a robotic handler with a articulated arm. It worked with objects laid down on a table and would made the assembly, placing the objects in their final positions

and orientation. Two computerized vision systems were developed (VISION using a 2D approach and LASER, a 3D system) capable of recognizing and identifying the position and orientation of the objects.

The agents integrating the multi-agent system were:

- WD (World Descriptor) - capable of establishing symbolic relationships between objects. It could, for instance, conclude that an object was on top of other;
- 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) - controls the robot. It is capable of geometric reasoning in order to materialize the symbolic operations issued by the TLP agent. It is a reactive agent and uses proximity sensors during task execution. Therefore it can react to unexpected situations like the obstacle detection, followed by mapping and obstacle remotion.
- MODELS - this agent stores several object models which are useful for creating symbolic relationships and the execution of assembly and manipulation tasks.

If we exclude LASER and VISION agents, that had an identical function, all the others were functionally different. The problem was then decomposed into subproblems to be solved by the agents. Figure 7 can give an idea of the flux or requests between agents during a typical task execution in the system CIARC.

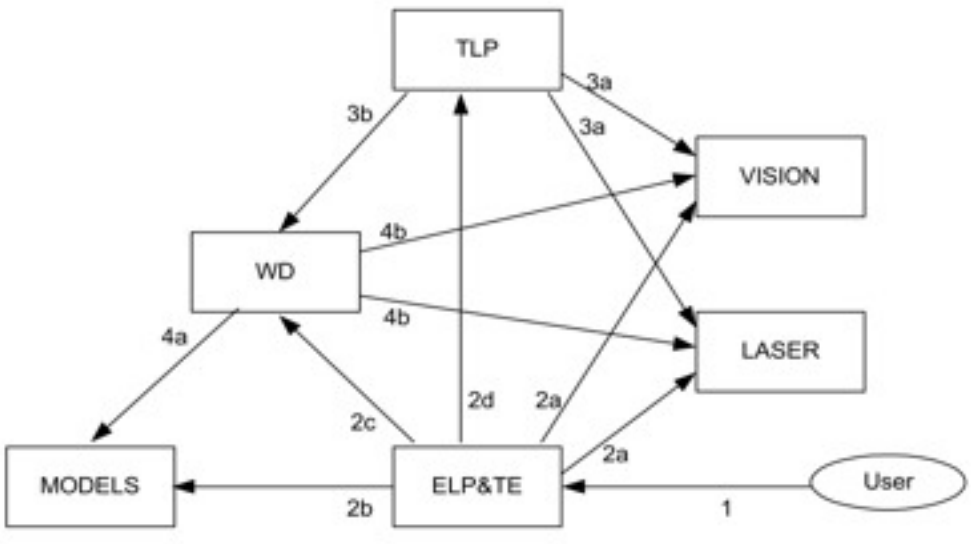

Figure 7 – CIARC: task execution requests

The user, that may be seen as another agent, makes a request to the agent ELP&TE for a certain assembly task to be performed (request 1). ELP&TE needs to know the objects' location (position and orientation) and asks that to agents VISION and LASER (request 2a). It also needs to know the manipulation models that it requests from MODELS (request 2b), the symbolic relationships and restrictions to be supplied by WD (request 2c) and the high-level symbolic plan by TLP (request 2d). Now TLP needs the objects' location from VISION and

LASER (request 3a) and the symbolic relationships and restrictions from WD (request 3b). WD will request the geometric models from MODELS (request 4a) and the objects' location from VISION and LASER (request 4).

We will now describe the holonic architecture to be applied to a Production System. In this architecture we will have holons representing tasks (HT), holons representing resources (HR) and others representing products (HP).

The basic task holons may get together to form task holons of greater dimension, or instead, a task holon can be decomposed in several more basic task holons. A task to make 10 sets of i table and 4 chairs may be decomposed in 2 task holons, one to make 10 tables and another to make 40 chairs.

Likewise, resource holons may be decomposed or be part of other holons. let us consider, for instance, a production cell with a lathe, a milling machine and a robot. The production cell may be a holon with 3 holons as components (lathe, milling machine and robot). That cell may be inserted in a production line holon, made of several cells. Several production line holons may be part of a factory holon.

Something similar may be seen with the product holons. In this case, they may be composed of component holons (themselves also being products).

The various phases of the production process may involve different holons. The scheduler holon (HEsc), for instance, is composed of task holons and resource holons and the process planning holon (HPP) is composed of resource holons and product holons.

A holonic organization can be defined as having a holarchic structure (holarchy) like an organization can be said as having a hierarchic structure. The holarchy is going to define the cooperation style, subjecting the holons to pre-defined goals and limiting their autonomy (without this restriction we will have something like an anarchic system).

Figure 8 presents an example of a holonic architecture applied to productive systems.

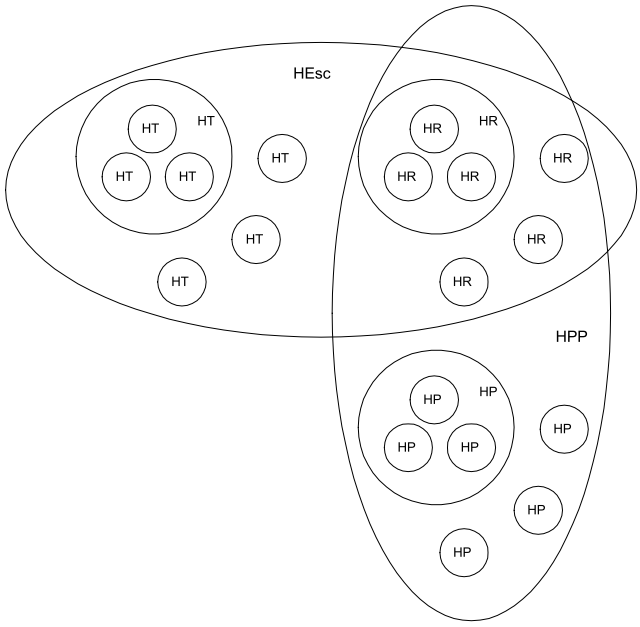

Fig. 8 – Production system's holonic architecture

7. Agents and Multi-Agent Systems support services

An infrastructure to support Agents and Multi-Agent System should provide a set of services that will exempt the developer from the obligation to build a system from scratch, letting them concentrate in the modeling, project and implementation of the agents.

This section will deal with the main support systems. They can be classified as follows:

- Communications;
- Security
- Directory or Informations
- Conversation

The main bibliographic source for this section is [Silva-1998].

7.1. Communications

We will discuss here several basic aspects like message exchange, synchronization, pooling and forwarding.

7.1.1. Message exchange

A basic aspect of communications in Multi-Agent System is the message exchange. We will now refer the main message exchange mechanisms.

7.1.1.1. Point to point message exchange

The Point to point communication is based on a bilateral message exchange. The message is sent by an agent and received by another but no other agents will know about it.

In a multi-agent system this type of communication occurs when the agent knows to whom it wants to talk to in order to get or supply some kind of information.

7.1.1.2. Group message exchange

In this case an emitter agent will send a message to a group of receiving agents. This method is also known as multi-cast.

In a multi-agent system this type of communication may occur when an agent have a list of agents to whom may send or request information.

7.1.1.3. Broadcast message exchange

The Broadcast method consists of a non directed dispatch of messages. All the community members will receive the message that has been sent. Typically there is no assurance though that all the potential receivers will get the broadcasted message.

In multi-agent systems the broadcasted messages are used when an agent doesn't know to whom exactly the message should be sent or when it wants to notify ll the community about something.

7.1.1.4. Blackboard

The Blackboard concept is rather different from the conventional message exchange. In this case, instead of using message queues or sockets, it used an area of shared memory. It was one of the first mechanisms to support multi-agent systems.

Every agent in a multi-agent system has access to this shared memory area and communicate through it. The Blackboard is the place where agents post their requests and replies. Any agent may accept requests that have been announced there or freely use the answers given or any information posted.

The implementation of this mechanism will be easier when all the agents are in the same computational resource because they can directly access the same memory area but is more complex when the various agents are spread over several machines. In this case it is common to use a server that encapsulates the Blackboard mechanism.

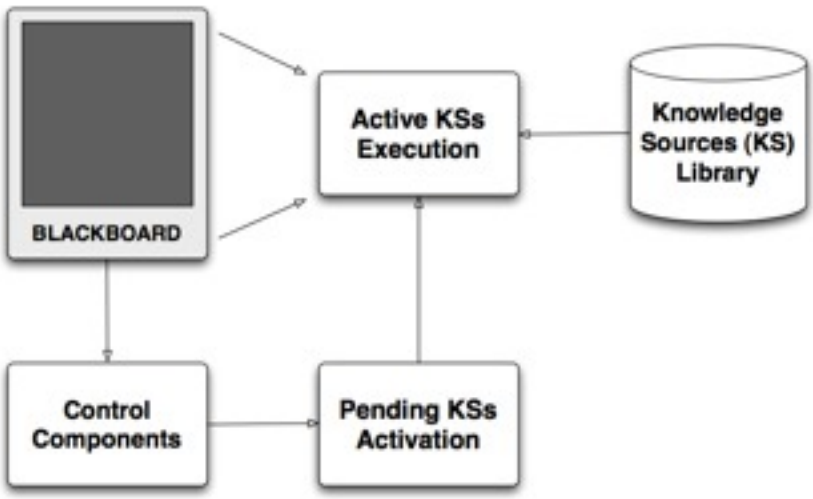

Fig. 9 - A typical blackboard architecture [Engelmore, 1988]

The exact mechanism, as depicted in the Figure 9, can be described as being composed of:

- a set of independent entities called knowledge sources (KS) that have specialized knowledge;
- a shared data structure, called blackboard, that the knowledge sources use to communicate.

The best way to describe the way these systems work is by using the following metaphor [Engelmore, 1988]:

> "A group of specialists are seated in a room with a large blackboard. The specialists are 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."

In a similar fashion, knowledge sources will be able to read and write in the central data structure. The problem solving process will take place with every KS monitoring the blackboard and presenting its contributions whenever finds a solution to the sub-problems that have been posted there.

2. Synchronism

There are two types of communication in what concerns the wait for a reply: synchronous and asynchronous communication

2.1. Synchronous communication

Synchronous communication is a message exchange protocol in which the receiver expects the arrival of a message in a pre-determined moment. The receiver therefore stops its processing while waits for the arrival of the message but that may happen quickly or take longer.

This protocol has several shortcomings: while waits, consuming computational resources, the agent doesn't produce any useful work. On the other side, we should consider the possibility that the message it is waiting never arrives (because the the other agent couldn't deliver the requested results or because some communication breakdown occurred). If any of this happens the agent will stay frozen indefinitely.

2.2. Asynchronous communication

In the asynchronous communication the communication and processing abilities are independent, i.e., the internal processing will not be affected by the message reception so that when a message arrives the agent will be notified and the message stored. Upon reception, the messages will be stored in a queue for future processing.

Additionally, the communication system may also detect the message type and fire automatically the correspondent processing functions.

This mechanism is useful due to the uncertainty of processes because one doesn't know when a message will be sent.

3. Pooling

Unfortunately, the reliability of communications is not always as it should be. For whatever reasons, breakdowns may occur. Therefore multi-agent system must have mechanisms that give them some capacity to tolerate faults. That's why the pooling service is used. When an agent stays inactive for an unforeseen reason, the pooling system will store the messages and

other relevant information waiting for the agent as soon as it is recovered. This way we assure that some recovery capacity is maintained.

4. Forwarding

It is not always possible for an agent to communicate easily with another agent within the community. In those situations it should be provided a forwarding service that receives the information the agent wants to send and takes care that it will arrive to the intended recipient.

7.2. Security

The security requirements exist in any system, namely in the Distributed Systems, where there is no single centralized entity responsible for keeping the consistency, objectivity and validity of its components' intentions. These systems are dynamic both in the structural and the organizational sense, being composed of distributed, autonomous entities but that are collectively committed to global purposes.

The system's security level is equal to smaller security level of all the system components. Therefore, the system's security must be addressed as a whole, both at communications and at the information levels.

In a multi-agent community the it may happen that someone may try, using an agent, to register into the system and access the community without permission. A first identification validation could be made with username and password as in any conventional system. To protect the flow of information from prying eyes a secure encryption standard can be used.

7.2.1. Names services

As a Multi-Agent System is composed of distributed and autonomous entities, the agents' heterogeneity and different goals may threaten the system consistency and objectivity. The system is highly dependent on each entity's intentions, its social behaviors and the way it abides by a basic set of protocols.

From the communications standpoint it will be necessary that, prior to any socialization, an entity is subject to a basic verification process, forcing the agent registration, my means of username/password validation. Each agent will supply its identification to the names service that stores and verifies the agents data. Only after this process the agent will be granted permission to participate in the agents community.

7.2.2. Message encryption service

The information circulating in the network is sometimes critical, be it classified information , credit card data, transactions or passwords. The message encryption service increases the security level of the information exchange between agents in a multi-agent system.

7.2.3. Permissions service

The fact that an agent is part of a multi-agent community doesn't confer automatically the rights to do access all kind of information or resources. In fact, the agent may be restricted its access to certain types of information, the use of certain resources and the requests of certain services.

The permission service will define “who has access to what”. It is common to divide the agents in groups and grant a specific set of rights to each group, like operating systems do with their users.

7.3. Information or Directory services

The Information or Directory service informs an agent about which other agents are capable of performing a certain task and how these agents may be contacted.

This information could exist in the agent itself but this solution would compromise the dynamic character and the robustness of the system. If a new agent, with a certain ability, is added to the multi-agent system, it is enough that it registers itself in the Directory service in order that this information is available for every agent already belonging to the community.

It is possible to draw some analogy between the Directory service and the phone companies’ Yellow Pages, the YP (Yellow Pages) service offered by some computational and operating systems, the NIS (Network Information Service), the NetBI or X.500.

The Information or Directory service can assume one of these two forms:

- Facilitator - it is responsible for the publishing and distributing information about the services or tasks that each agent is capable to perform as well as its contacts.
- Broker (Discovery) - will search, when requested, for some particular information, using other information services or questioning other agents about their capabilities.

7.4. Conversations

A conversation is an ordered set, not necessarily sequential, of messages transmitted between two or more agents, that are mutually understood by the intervening entities. The conversation systematizes and defines the occurrences and the types of messages in such a way that, in a certain moment of the conversation, the set of possible messages is limited, mutually understood and correctly used.

The type of conversations can go from a simple exchange of control numeric values to complex negotiations, with uncertain outcomes and involving multiple interactions and intervening entities. The actual needs of current systems lay somewhere between those two extremes.

Conversation requires control and monitoring. Some of the mechanisms that are used will be briefly referred in the next sections.

7.4.1. Time-out mechanism

An agent cannot wait indefinitely for an answer from another agent. The time that one agent should wait for an answer is called timeout. This timeout is not fixed and should depend on the conversation state and on the intervening agents. This is the reason why this issue was not considered in the communications component.

7.4.2. Information Management

The conversation related information must be adequately stored. The conversation state, the intervening agents and previous messages data are usual elements in any conversation. Therefore, it makes sense that this data is managed by the specific conversation process and not by the application itself.

7.4.3. Synchronization

The conversation synchronization consists of the set of control activities that manage the time and order of messages processing.

A certain conversation is composed by a set of ordered messages but if a particular order is not followed it shouldn't be automatically a cause for fatal error and reason for the conversation to be aborted. Situations may occur in which the messages arrival occurs before the moment expected by the conversation. Nevertheless, if those messages are stored and made available at the correct moment, the conversation will be able to proceed without problems.

8. Negotiation between agents

As in the negotiation between humans, the agents must be able to communicate between themselves and to exhibit some social abilities in order to establish a negotiation.

In order to control the development of the negotiation process between agents some negotiation protocols were defined. The more common types of negotiation are:

- 1 potential contracting entity to 1 potential contractor;
- 1 potential contracting entity to N potential contractors;
- N potential contracting entities to 1 potential contractor;
- N potential contracting entities to N potential contractors.

We will discuss the three first types of negotiation because the last one can be obtained by the product of the second and third cases.

8.1. One to one negotiation

We can see the Client-Server relationship as a particular case of this type of negotiation. The difference lays on the fact that the potential contractor, unlike the server, may refuse to perform the required tasks.

The protocol starts with an announcement (Figure 10) made by the contracting party (Ag) to the potential contractor (Ag1).

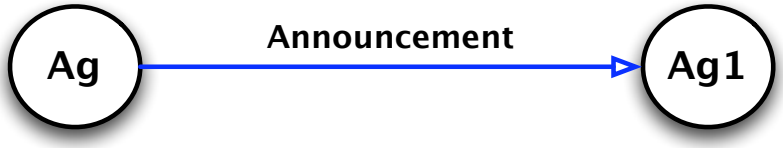

Fig. 10 – Announcement from Ag to Ag1

The agent Ag1 will then analyze Ag's announcement and will answer with a proposal or saying that it cannot execute the required task (Figure 11).

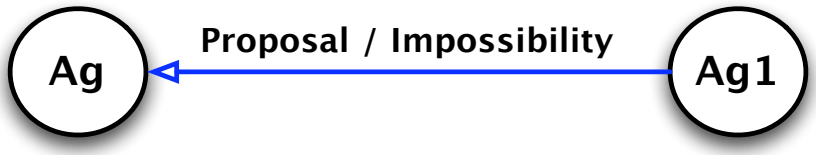

Fig. 11 – Ag1's answer to Ag

In the last case, Ag knows that cannot count on Ag1 to perform the desired task. Ag should be sufficiently autonomous to know how to deal with the situation, avoiding an impasse. If Ag receives a proposal from Ag1, it will analyze it and decide whether to accept it or reject it (Figure 12), depending on the proposal's contents.

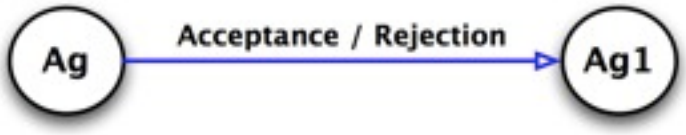

Fig. 12 – Finalising the negotiation between Ag and Ag1

The process can grow more complex and demand more interaction between Ag and Ag1. Ag may, for instance, set a maximum price or a delivery date and Ag1 may try to convince Ag to accept a proposal that fits the required price but not the delivery date, or the opposite.

Agents should have fault tolerance mechanisms, namely in what concerns communications, because a message may not reach its intended destiny for a number of reasons. Again the issue of agent's autonomy appears as been a key issue.

8.2. One to N negotiation

Several negotiation protocols of one contracting entity to N potential contractors have been proposed, usually based on the "Contract Net Protocol" [Davis-1983].

Let's consider an agent Ag willing to get a certain task done. This task may be performed by any of the potential contractors, the agents Ag1, Ag2 and Ag3.

The process begins with the announcement made by Ag and directed to Ag1, Ag2 and Ag3 (Figure 13).

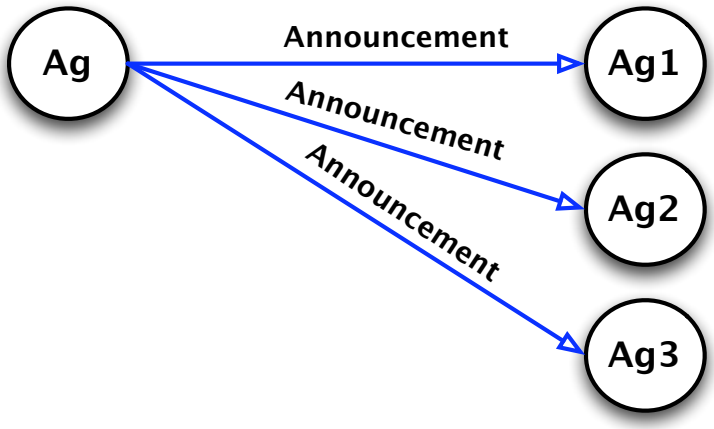

Fig. 13 – Announcement from Ag to Ag1, Ag2 and Ag3

In order to make the announcement, the agent Ag must know who is able to perform a certain task (meta-knowledge about other agents' abilities). In alternative, it could get this information from a specialized agent, to issue a broadcast or to use a shared memory area to post the announcements (blackboard).

The agents Ag1, Ag2 and Ag3 will then analyze the announcement and react, sending proposals, declaring themselves unable to satisfy the request or simply not answering (Figure 14).

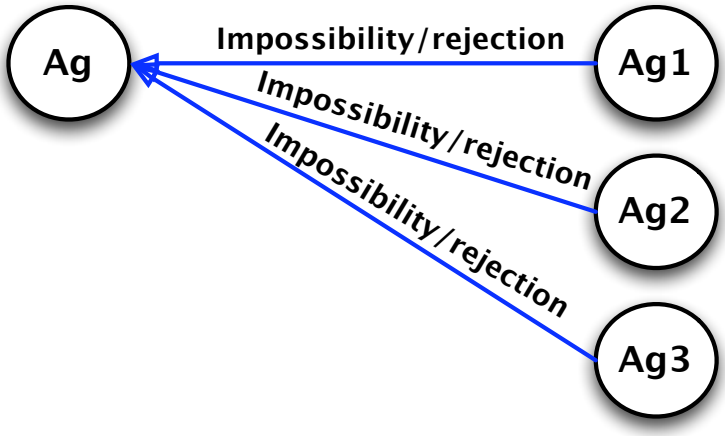

Fig. 14 – Ag1, Ag2 and Ag3 answer Ag

Ag analyses the received proposals and selects the best one among the ones that fit the requirements, informing the corresponding agent about its acceptance. It may eventually inform the agents whose proposals were turned down about its rejection (Figure 15).

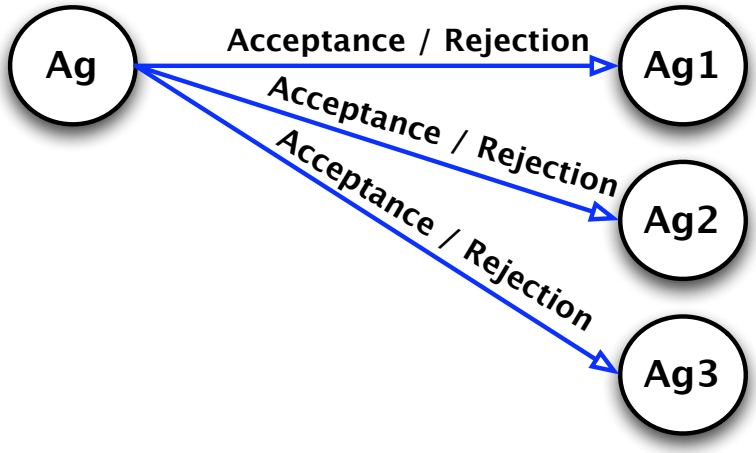

Fig. 15 – End of the negotiation process

When the impossibility or rejection messages are not mandatory, the protocol should take this into consideration, otherwise the agents risk waiting for an event that will not ever occur. Even when these messages are obligatory it can happen that they don't arrive due to failure in the communications process. Therefore, the agents must be prepared to be tolerant to this kind of faults (the timeout mechanism can be a solution).

Let's consider the following problem:

Agent Ag wants 3 tasks (T1, T2 and T3) to be performed. These tasks have the following restrictions:

T1 must precede T2
T2 must precede T3
T3 must be concluded before the instant 10 (10 time units)

The tasks may be performed by the following agents:

T1 may be executed by Ag1 or Ag3, its duration being 2 time units
T2 may be executed by Ag2 or Ag3, its duration being 2 time units
T3 may be executed by Ag3 or Ag4, its duration being 3 time units

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.

The problem above requires several negotiations of 1 to N (3 in this case), meaning a negotiation for each task (T1, T2 and T3). It should be noticed that those negotiations are inter-dependent. The agent Ag3, for instance, may execute the T1 and T2 tasks. It may happen that this agent will be awarded the execution of both tasks. Therefore, the negotiation of T2 may be influenced by the negotiation of T1. This fact may bring some problems to the simultaneous negotiation of tasks.

In order to solve the problem let's assume that we have 3 protocols of 1 to N, associated with T1, T2 and T3, to deal with. We will only start negotiating T2 when T1 is finished. Likewise, the T3 negotiation will only start when T2's is completed.

In the negotiation of T1, the agent Ag contacts the potential contractors, Ag1 and Ag3, asking them to execute the task T1. On the other hand, let us suppose that Ag knows that those tasks require a total of 7 time units to be performed (2 for T1, 2 for T2 and 3 for T3). Therefore, the deadline for concluding the T1 task will be the instant 5 (10 - 2 - 3), because 10 time units is the alloted time period to execute the whole set of tasks and the remaining tasks (T2 and T3) take 2 and 3 time units to perform, respectively.

It may happen, though, that Ag doesn't know the exact durations of these tasks. It is actually more natural that this information is known only to the contractor agents. Moreover, the duration of certain tasks may not be known beforehand, forcing the agents to work with estimative values.

Let us suppose that, according to their agendas, Ag1 and Ag3 both answer to Ag saying that they can execute the task T1. The possible intervals for Ag1 and Ag3 are:

(2,4) for Ag1

(3,5) for Ag3.

Ag has now to choose between them and it will probably choose Ag1 because this agent can start sooner. However, the choice could be different if it followed a JIT (Just in Time) methodology or if it was dealing with a cost sensitive application and Ag1 had higher costs.

Now Ag starts the negotiation the task T2. The available timeframe is 7 (10 - 3) and the potential contractors are Ag2 and Ag3. No need obviously to start T2 before the instant 4 which corresponds to the end of T1. Let us also admit that there are no minimal waiting periods between T1 completion and the start of T2. The opportunity window for the execution of T2 is then the interval (4,7).

According to their agendas, only Ag2 can execute T2 during the following interval:

(4,6) for Ag2.

Finally, the T3 negotiation starts. The opportunity window for the execution of this task lays between 6 (the end of T2) and 10 (final deadline). The potential contractors are Ag3 and Ag4.

The sole agent capable of performing this task with the imposed restrictions is Ag4 and the interval will be:

(6,9) for Ag4.

The negotiation process ends successfully. It could happen though that in order to fulfill all the time limits the process would need to backtrack, that is, to go back and redo the first steps of the negotiation trying to get a different solution. It could even occur that no solution could be found for the problem in such a way that all the restrictions were met. In this case, we could try and relax some of the restrictions.

8.3. N to One negotiation

It is not really a special type of negotiation because it is the result of several contract processes like the previous one but now centered on the contractor point of view. Now, the potential contractor has do deal simultaneously with the negotiation of several contracts.

Let us consider that three agents (Ag1, Ag2 e Ag3) want to close a contract independently with Ag. Therefore, Ag receives 3 different announcements (Figure 16).

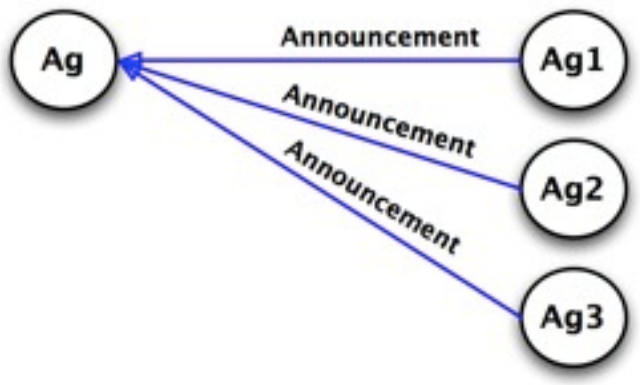

Fig. 16 – Announcements from Ag1, Ag2 e Ag3 to Ag

The question that the potential contractor (Ag) must address is how it can commit itself with each of the proposals it sends if it is not sure about any of them being accepted by the receivers. In fact, Ag1, Ag2 and Ag3 may reject its proposals or even being at the same time in a negotiation process with other agents. Let us illustrate the problem with an example:

Suppose that Ag receives

> *An announcement from Ag1 to perform the task T1 with a 3 time units duration and to be finished by instant 4*
>
> *An announcement from Ag2 to perform the task T2 with a 4 time units duration and to be finished by instant 6*
>
> *An announcement from Ag3 to perform the task T3 with a 2 time units duration and to be finished by instant 5*

Ag agenda is completely free.

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

Ag will be able to individually satisfy any of the three request (T1, T2 or T3). A brief analysis may find that he also could take care of the pairs T1-T3 or T3-T2. We can conclude then that the agent will be able in any case to execute T3 and will then have to choose between T1 and T2. Let us suppose that it chooses T2 because it is longer to execute and therefore potentially more rewarding. Ag will then answer Ag2 and Ag3 with accepting proposals and rejects Ag1 task. It may happen, though, that Ag2 and Ag3 choose another agent to perform T2 and T3 and therefore Ag ends up loosing everything. It could even happen that Ag1 doesn't have any alternative proposal for T1. We could say then that Ag had made unhappy choices regarding these announcements. Of course, 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.

We are in the presence of a *Indecision Problem*, that may be stated as follows: *When an agent is subject to the simultaneous negotiation of several contracts it will never know if its proposals will be accepted or turned down, and that will affect its behavior*.

An optimist and reliable agent understands that when it sends a proposal it may be accepted and, therefore, in subsequent negotiations it will consider as not available everything that has been object of a prior proposal (time, goods or services). Such an agent will tend not to be available for other negotiations. However, it risks that if its proposals are refused, it will have lost additional opportunities to close a deal because it was not available for other negotiations.

A pessimist and unreliable agent knows that there is no assurance that its proposals will be accepted and therefore in future negotiations considers that what has been included in prior proposals is still available, due to the fact that those proposals have not yet been accepted. This agent will make a better use of the opportunities but risks the discredit because the other agents ay start considering its proposals as not credible (if they are able to learn from experience...).

Some of the alternatives solutions for dealing with this problem are:

- To make the negotiations one by one until the end in such a way that only after receiving an acceptance or a rejection one starts negotiating the next proposal (will it be realistic under normal conditions?);
- To include additional steps in the negotiation protocol indicating that the proposals need to be reconfirmed by the contractor after being selected by the contracting party;
- To evaluate the relative importance of the potential contracts taking into account who the customer (contracting entity) is, and using heuristics;
- To gauge the impact of the non-fulfillment of contracts (penalties);
- To try and subcontract other agents for the overload;
- To try and renegotiate contracts already committed.

In order to be able to deal with this type of problems an agent must be provided with enough “intelligence” and knowledge, specially when the agent is in the contractor position.

There are several characteristics that can be associated with an agent’s character. You can, for instance, describe an agent as reliable or unreliable, optimistic or pessimistic.

Finally, one should note a characteristic like the learning ability is very important in order that the potential contractors don’t find themselves repeatedly in a loosing situation.

8.4. Renegotiation

The renegotiation may be seen as a new negotiation process that is started because a previously established negotiation has been broken. Imagine that in the example given on section 8.2 (a 1 to N negotiation) the agent Ag selects Ag1 as a contractor and therefore sends Ag2 and Ag3 rejection messages. If Ag1, the awarded contractor, fails to deliver, when Ag becomes aware of the fact sends new announcements and repeats the previously described negotiation process, this time with agents Ag2 and Ag3 only.

9. Auctions

This is a very common type of negotiation, frequently organized for selling art objects and often reaching values much higher than initially foreseen. Auctions can be used for selling individual items, like paintings, or sets of itms like Treasure Bonds.

Auctions are quite appropriate in situations where the goods don't have a fixed or pre-determined market value or, in other words, when the seller is not sure about what price to ask. Auctions are specially useful in countries that are beginning to adopt a market economy because they make possible the evaluation of the goods' value when no prior estimation exists.

It is more flexible to sell something in an auction than to fix previously the selling price and can be quicker and less costly than negotiating a price. When one negotiates a price, each proposal or counter-proposal is considered separately but, in an auction, the competitive proposals are made almost simultaneously.

An auction is a method for assets' attribution, a method intrinsically based on competition. It is the purest of the markets: a seller wants to get the biggest profit and the seller wants to pay the lowest price. An auction offers a simple way of establishing the price based on the market. It is an efficient mechanism in the sense that an auction usually assures that the resources are alloted to the one that values them the most and that the sellers receive the value given by the item's evaluation.

A particular characteristic of this method is that the price is not defined by the sellers but by the bidders. However it is the seller who defines the rules when he chooses the type of auction to be used. Another peculiarity of the auctions is that it is not usually the auctioneer that keeps the goods but instead acts as an agent for someone that wants to buy them.

Frequently, the buyers know more about the product's value than the sellers themselves. The sellers quite often don't suggest a price because they fear that their ignorance may cost them dearly, and promote an auction instead, asa way of gathering price information that otherwise would be tough to get.

There are several ways of classifying auctions. There are open auctions and auctions with secret proposals. There are auctions where the price increases with every bidding and others in which the price decreases at regular intervals. There are unilateral and bilateral auctions. In the unilateral auctions only the buyers can make proposals whereas in the bilateral auctions both sellers and buyers are able to make them. The unilateral auctions can also be based on offers made by sellers. This type tend to favor the silent entities, like buyers waiting for the proposals to go below the competitive equilibrium price or sellers waiting for the offers to go above it.

An important remark to be made deals with the kind of motivations to participate in auctions. The obvious one is the acquisition of goods for personal use or consumption. In this case the object to be auctioned has a value which is specific to each bidder. A different motivation is to acquire goods for resale or commercial use. In this case, the proposal's value depends not only on the private evaluation but also on future evaluation of the next buyers. Each bidder

tries to estimate the final price of the product. The people's behavior will then vary according to the type of motivation.

William Vickrey [Agorics - URL] established a basic taxonomy for auctions based on the way prices are quoted and the bids are dealt with. Four types of standard unilateral auctions were defined: the *English Auction*, the *Dutch auction*, the *First price sealed-bid auction*, and the *Second price sealed-bid auction*. The table II shows the basic rules and how the final price is set in these four auction types.

Auction type	**Rules**	**Closing Price**
English (open, oral, ascending)	Seller may set a "reserve" price. Bidding price increases until there is 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.

Table II – Rules and price setting in auctions

Agents can also use auctions in their negotiations, with seller agents and buyer agents (bidders). These agents must show intelligence and even cunning in order to participate successfully in auctions.

Certain newly emergent markets previously controlled by monopolies, like the Electricity Market, may be implemented using an auction mechanism. In these markets, we will have producers and buyers of electrical energy. It makes sense that these entities are represented by agents. In the internet, there are many products or services (travel, for example) that are sold through auctions and the agents' presence is more and more evident.

A better description of the several types of auctions can be found in the reference [Praça-2001] where this section was based.

10. Conflicts

Conflicts may occur in a Multi-Agent System as it happens in any conventional society. We have already described the main types of conflicts:

- Conflicts of goals - agents' goals are not the same, they may even be contradictory.
- Conflicts of Responsibility or Interests - there are several agents willing to take responsibility for a certain task or to fulfill the same goal.
- Conflicts of Information or Knowledge - several agents have different views on the same situation or reality.

The first type of conflicts (of goals) occurs when two or more agents analyze a problem from different, eventually contradictory, perspectives. Let's consider the following example:

An informatics department will purchase a certain equipment based on the recommendations submitted by two different agents: one agent makes the economic evaluation while the other is responsible for the technical one. The agents receive the suppliers' proposals, analyze them and issue a recommendation about the equipment to be bought. Minimum specifications (CPU, amount of memory, display, disk size, etc) will be defined and a maximum budget will be estimated.

The agent responsible for the economic evaluation will tend to opt for the computer that, satisfying the minimum specifications, has the lower price. The agent that takes care of the technical evaluation will prefer the computer with better characteristics provided that its price isn't higher than the maximum budget. A conflict arises because its goals are different (economic versus technical). The agents performance could obviously benefit if they could argue about the different solutions found and could establish a dialogue with potential suppliers, trying to convince them to decrease prices or offer better characteristics. Agents should be able to reach compromises in order to assure that at least the most important part of their goals is met.

The fact that the agents' goals are different doesn't necessarily imply that they are contradictory. However, if we had choose between different mutually exclusive policies to be applied , the situation would be diverse. Let us consider, for instance, an airplane controlled autonomously. Its mission is to make vigilance over a certain area. It starts starts from an certain airport and its work is carried out close to another airport. Suppose that the weather in the area it is investigating is getting worse and a choice has to be made: to return to the airport from where it lifted off or to go on and try to land on the other airport, with the risk of running out of fuel, because it goes beyond the point of no-return. If we consider two agents responsible for the control of the airplane, one whose goal is to get the job done s quickly as possible and the other more concerned with security issues, a conflictual situation will be reached, because their goals are, in the current situation, contradictory. Only one of the goals may be pursued. Therefore, the system must have a priorities scheme that chooses the goal of the agent with the higher priority whenever the consensus is not achievable.

In the responsibility or interests conflicts, the agents' individual goals can even be the same but, as they are competing for a single and non-divisible asset, only one of the gas can be satisfied. A typical example is the bid for a single product in an auction. The competing agents

have a similar goal but will never be able to cooperate because only one can win. A similar situation occurs when several companies sell the same range of products. In order to deal with this situation, the agents must be able to foresee the actions of their competitors or even spy on them (an agent could, for instance, check other agents' prices before defining their own) .

Finally, we have the conflicts that arise due to differences of data, information and knowledge. Several agents can have a different view of the same reality. One example is the object identification systems VISION and LASER, described on Section 6.2, that could consider that in a certain position existed two different objects. This problem can be solved by attributing different levels of credibility to the different agents, in what concerns the information they supply in a given situation. Other possibility will be to try merging the data/ information/knowledge from the various agents looking into the same reality in order to try and use part of the results produced by each agent. It will be necessary to define the data/ information/knowledge items to be treated and check whether all the agents supply them. If only one agent supplies one certain data item we will be forced to accept it as true. If several agents agree in a certain item then its credibility will increase. If there are contradictory itms we will have to check the credibility of each of the agents supplying it. We can further elaborate the process if we add to the agent credibility rating a measure of certainty in its conclusions. An agent we believe in may be only 60% sure of a certain data item being true while another agent in which we believe less may be 9% sure about the same data item.

11. Interaction between Agents

11.1. Ontologies

In order that two agents are able to communicate and understand each other they need to be able to use a common language or, at least, to use languages that are mutually translatable. But this is not enough for any two agents to understand each other. They also need to share a certain knowledge organization or, in different words, they need to share an Ontology.

An Ontology specifies a representation vocabulary for a specific domain from which the words used by the agents will be chosen. But the Ontology is more than a vocabulary containing concepts because it treats these concepts at a higher level, defining classes, relationships and functions.

The term Ontology has caused some controversy in the Artificial Intelligence field and other areas but it has been used since long time ago in Philosophy to describe the “theme of the existence”, the nature of being. Sometimes Ontology is confused with Epistemology which is the theory of knowledge.

Tom Gruber defines Ontology within the scope of knowledge sharing as being a “specification of a conceptualization”. In this sense, an Ontology is seen as a specification of the concepts and relationships between them that an agent or an agents community may use in their communication, as it happens in the formal specification of a computational program [Gruber-1993].

Ontologies have become increasingly important in the scientific communities related to Knowledge Management, Intelligent Agents and Multi-Agent Systems. In Knowledge Management, the importance of the ontologies comes from the need to share and reuse knowledge within an organization. In the area of Intelligent Agents and Multi-Agent Systems, Ontologies are key to allow the agents to establish a common platform on which to base the knowledge interchange. Without it, each agent risk interpreting differently the same terms. In the limit, we will produce *Idealects* with agents emphasizing strictly private perspectives on concepts that should be universal.

Therefore, it is important to define ontological commitments between agents. An ontological commitment is an agreement between a set of agents in order to use a common vocabulary in a way that is consistent with the theory specified by the ontology.

An Ontology can be represented by a Knowledge Base hierarchically structured in classes. However, this kind of structure is not mandatory.

There are several criteria that should be observed when designing a new ontology:

- *Clarity* - an ontology should describe in an efficient manner the meaning of the terms that are being defined. The definitions should be objective. Whenever possible the definitions should be complete (by means on necessary and sufficient conditions) instead of partial;
- *Coherence* - if a sentence is inferred from the axioms on an ontology and contradicts a definition present in that ontology then this ontology is incoherent;

- *Extensibility* - it should be possible to define new terms for special uses based on the vocabulary without altering the existent definitions;
- *Minimal codification* - the conceptualization should be done at the knowledge level, without specific coding at the symbols level. It should be noted that the agents may be implemented with different styles or representation systems;
- *Minimum need of logic commitments* - due to the fact that ontological commitments are based on the consistent use of the vocabulary, these commitments may be minimized by using the weaker theory (the one that fits all theories) and defining only the themes that are key to the communication of consistent knowledge according to that theory.

Ontolingua is a system that allows for the definition of ontologies that are portable between several representation systems. Other alternatives are KADS, Conceptual Graphs, IDEF5 e BSDM.

11.2. Knowledge Exchange Formats

KIF (Knowledge Interchange Format) defines a format to be used in the knowledge exchange between agents. It has a declarative semantics, the meaning of expressions can be understood without resorting to an interpreter capable of manipulating those expressions.

KIF is logically understandable. With it, one can express realities using first order logic. KIF may be seen as a prefix version of 1st order predicate calculus, with several extensions to increase its expressivity.

KIF allows to represent meta-knowledge.

Examples of KIF messages' contents may be:

- Notification: (tell (> 5 3))
- Request: (perform (print "Hello!" t)
- Answer: (reply available)
- Question: (ask-if (> (profit product_1) (profit product_2)))
- Subscription: (subscribe (coordinates ?x ?y ?z))

The next example shows how to express the idea that a certain worker, belonging to a certain department and identified by his fiscal number, has a specified salary.

(salary 123456789 accounting 1500)
(salary 132547698 purchasing 1200)
(salary 143276598 marketing 1800)

It is also possible to establish comparisons and to make calculations. For instance, to compare the area of two tracts of land, we can use:

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

We can express that the even power of a real number is greater than zero by stating that:

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

The use of the operator ‘ and the commas allow the agent to say that it is interested in receiving the 3 values associated with the salary relationship:

(interested joe ‘ (salary ,?x ,?y ,?z))

KIF can also be used to describe agents’ programs or scripts. Due to the prefix nature of KIF’s syntax, those programs look like Lisp or Scheme programs, as can be seen by the following code fragment:

(progn (fresh-line t) (print “Hello”) (fresh-line t))

11.1. Knowledge Query and Manipulation Languages

KQML (Knowledge Query and Manipulation Language) is a protocol for the exchange of information and knowledge between agents. It is a declarative language. It was proposed by a project called Knowledge-Sharing Effort of the US Defense Advanced Research Projects Agency in the late 80’s [Labrou-1988].

The KQML language has three layers:

- *Content*: an expression in an agreed format like, for instance, KIF (Knowledge Interchange Format);
- *Message*: expresses the communication logic, stating the language used to convey the content, the type of content expression;
- *Communication*: defines how the communication will be established, says who is the sender and the receiver, the type of communication and other related details.

KQML is indifferent to the content and format of the information and knowledge it handles. It ignores the message content except to determine where it ends.

The KQML language is composed of several reserved primitives (“performatives”). Some examples of these basic primitives are:

- Informative: tell, achieve, untell, unachieved;
- Requests: perform;
- Questions: ask-if, ask-one, ask-all, evaluate;
- Answers: reply, sorry;
- Capability definition: subscribe, monitor, advertise, unsubscribe;
- Communications: forward, register, unregister, broadcast, route.

A complete KQML expression includes, besides the speech primitive, a list of paits attribute-value with elements such as:

:language <*language*>

:content <*content*>

:ontology <*ontology*>

:from <*agent_origin*>

:to <*agent_destination*>

It is possible to insert pre-conditions, post-conditions and final conditions into a ialogue between agents using KQML.

The following example shows how agent A can inform agent B that it believes in X, using pre-conditions (Pre(A) e Pre(B)), post-conditions (Post(A) e Post(B)) and final conditions (Completion):

tell (A, B, X)

Pre(A): BEL(A,X) ∧ KNOW(A, WANT(B, KNOW(B,S)))

Pre(B): INT(B,KNOW(B,S))

Post(A): KNOW(A, KNOW(B,BEL(A,X)))

Post(B): KNOW(B,BEL(A,X))

Completion: KNOW(B, BEL(A,X))

S can be BEL(B,X) or ¬BEL(B,X)).

Let's have a few more examples of dialogues between agents A and B. In the first example A tells B that 3 is greater than 2:

A to B: (tell (> 3 2))

In the second dialogue A asks B to write "Hello" and B confirms that it did it:

A to B: (perform (print "Hello!" t))

B to A: (reply done)

In the third dialogue A questions B whether p1 product's cost is greater than p2's and B answers affirmatively:

A to B: (ask-if (> (cost p1) (cost p2)))

B to A: (reply true)

At last, agent A requests agent B to inform him about previous day minimum and maximum temperatures in several cities, as soon as they are available:

A to B: (subscribe (temperature ?a ?b ?c))

B to A: (tell (temperature porto 12 20))

B to A: (tell (temperature lisboa 13 23))

B to A: (tell (temperature faro 15 23))

A to B: (unsubscribe (temperature ?a ?b ?c))

11.4. Languages for communication between agents

The standardization of communication languages makes software interoperability easier because it frees the interface from the implementation and translation tasks. Standardization is becoming common in many areas like SMTP in email, GIF and JPEG graphical formats and Postscript in text formating programs and printers.

An agent communication language (ACL) gives the agents the means to exchange information and knowledge.

The scientific community of Intelligent Agents and Multi-Agent Systems approach the problem from two different perspectives: a procedural approach or a declarative one.

The procedural approach is based on the assumption that the communication between agents can be modeled by procedural directives. The script languages (like TCL, AppleScript or Telescript) are based on that approach, at the same time simple and powerful. It allows the transmission of individual commands or full programs. The disadvantages are typical ones for all procedural approaches. Sometimes, information about the receiver is needed that is not available at the sender side. Moreover, procedures are unidirectional when most of the information agents need to share must flow in both senses. Beside that, scripts are difficult to merge. This is not a problem in one to one conversations but when we have one agent talking with several others it may find difficult merging several scripts.

The conception of language used in the declarative approach considers that communication is better modeled by the exchange of declarations (definitions, assumptions and so on). The declarative language should be sufficiently expressive but compact, at the same time.

ACLs result from the work by KSE (Knowledge Sharing Effort) group from DARPA-DoD (Defense Advanced Research Projects Agency of the US Department of Defense) in the beginning of the 90's, using a declarative approach as its basis.

Also during this period, France Telecom developed Arcol, an ACL containing a group of primitives smaller than the one later defined in KQML. Arcol primitives, like KQML's are assertive or directive but, unlike them, primitives can be composed of others. Arcol has a formal syntax that assumes that agents have beliefs and intentions and that may represent their uncertainties about facts.

A widely known agent communication language is FIPA ACL (FIPA – Foundation for Intelligent Physical Agents – http://www.fipa.org/) that was strongly based on the Arcol model and semantics.

An ACL can be better described as being composed of three parts: the vocabulary, its "internal language" (KIF) and its "external language" (KQML). An ACL message is a KQML expression in which the arguments are terms or "phrases" in KIF composed with words chosen from an ACL vocabulary.

But what separates an ACL from technologies like RPC (Remote Procedure Call), RMI (Remote Method Invocation) or CORBA and object request brokers?

An ACL should:

- handle propositions, rules and actions instead of objects, implying that ACL worries about aspects related to semantics and meaning.
- describe the desired states in a declarative language instead of procedures or methods.

Agents not only exchange messages but they also establish conversations like, for instance, in a negotiation process. In ACLs it is common to refer "conversation acts" instead of exchanging messages. They can also be based in the BDI model (beliefs, desires, intentions).

Conversation acts can be divided in several categories. Some examples of those categories are:

- Assertive: the door is closed;
- Directive: to close the door;
- Questions: Is the door locked?;
- Commitments: I will close the door;
- Permissive: He can close the door;
- Prohibitive: He can't close the door;
- Declarative: This is the way out door;
- Expressive: I would like this one to be the way out door.

The dialogue can be based on propositional attitudes. A propositional attitude is a relationship between three parts (the agent, the proposition content and an element of a finite set of propositional attitudes). Let's consider this example:

Agent : Ag

Contents: interest rate decreasing

Set of propositional attitudes: {believes, wishes, imposes, recommends, avoids}

One propositional attitude could be:

<Ag, believes, interest_rate_decreasing >

This indicates that agent Ag believes that some interest rate decreasing will occur.

13 – Aplicações de Agentes e Sistemas Multi-Agente

Nesta secção iremos referir algumas áreas e tipos de problemas tratados pelos Agentes e Sistemas Multi-Agente.

13.1 – Comércio Electrónico

O Jango (http://www.jango.com/) da Excite o Bargainfinder (http://bf.cstar.ac.com) da Andersen Consulting são exemplos de agentes que pesquisam na Internet à procura do melhor preço para um dado produto. Também podemos considerar agentes que operam em negócios entre empresas, B2B, tais como o FairMarket e no mercado de derivados, como o ETrade e o OptiMark.

O Michigan Internet AuctionBot [Wurman-1998] é um projecto do Laboratório de Inteligência Artificial da Universidade de Michigan, nos Estados Unidos da América. É visto como um serviço de recolha de informação, que recolhe ofertas, determina os preços resultantes (usando regras próprias de leilões) e notifica os participantes. Não suporta transações de nenhum tipo específico, contudo o AuctionBot dispõe de uma API que está disponível para uso.

O Fishermarket [Rodrígues-1998] é outra abordagem de recriação de leilões on-line através da tecnologia dos agentes. Originalmente o Fishermarket suportava o modelo holandês de leilões, mas agora já suporta os outros modelos mais conhecidos (inglês, primeira e segunda propostas).

O MAGMA [Collins-1998], posteriormente conhecido como MAGNET, foi desenvolvido na Universidade de Minnesota e permite o estabelecimento de negociações entre agentes.

O KASBAH é um sistema da AmEC Initiative [Chavez-1996] ligada ao Massachusetts Institute of Technology que assenta num leilão duplo. Actualmente o sistema recebe o nome de MarketMaker.

O Tete-a-Tete [Guttman-1998] é outro projecto do Massachusetts Institute of Technology que assenta no conceito de comparação de valores, ao invés de comparação de preços. É usada uma função multi-atributo de modo a representar melhor as necessidades dos utilizadores. A negociação consiste na resolução distribuída de um problema sujeito à satisfação de restrições.

O CASBA [Vetter-2000] é um projecto que visa o desenvolvimento de um mercado electrónico e que segue as seis etapas do modelo CBB (Consumer Buying Behaviour).

A proposta de um modelo para um mercado electrónico que considera o modelo CBB é também feita em [Viamonte-2000], tendo a particularidade de envolver técnicas de Data Mining no processo.

O MarketSpace é uma infraestrutura de mercado aberta e baseada em agentes desenvolvida pela Universidade de Uppsala e pela empresa de telecomunicações sueca Telia. É um mercado aberto no qual a pesquisa, negociação, contratos e interacção com os utilizadores é feita recerrendo a agentes.

O ODB (On-Line Dynamic Bargaining) é um sistema que tenta encontrar um compromisso entre o preço que o vendedor pretende e o preço que o potencial cliente pretende oferecer. O preço vai variando de modo a que comprador e vendedor assentem num preço comum ao longo de um processo iterativo no qual o preço do vendedor vai baixando e o do comprador vai subindo [Lin-2001].

13.2 – Sistemas de Produção

Os sistemas de produção caracterizam-se por um elevado grau de complexidade, sendo naturalmente distribuídos do ponto de vista físico (várias máquinas, linhas de fabrico,

fábricas, etc) e lógico (vários produtos, encomendas e ordens de fabrico a serem tratadas simultaneamente).

Aqui é costume encontrarmos Sistemas Multi-Agente e haver algum realce para a componente de negociação.

A cooperação e negociação podem ser vistas a diversos níveis, como por exemplo:
- Entre recursos de uma mesma linha de produção ou de uma mesma célula de fabrico;
- Entre diversas linhas de produção ou células de fabrico;
- Entre a empresa e os seus fornecedores ou clientes (empresa estendida);
- Entre diversas empresas para tirar partido de uma nova oportunidade de negócio (agilidade, empresa virtual).

O sistema YAMS foi talvez a primeira aplicação dos Sistemas Multi-Agente à área de sistemas de produção. Utiliza um protocolo de negociação entre agentes baseado no Contract Net [Parunak-1987].

A General Electric Power Generation utilizou o protocolo Contract Net no processo de negociação entre 35 estações de trabalho de um sistema job shop. O cliente faz pedidos de produtos, os agentes decompõem o produto em partes e fazem pedidos aos fornecedores e o processo repete-se até que a cadeia de fornecimento venha a emergir e os agentes podem desse modo enviar as suas propostas para os clientes que podem confirmá-las.

O FLAVORS é um sistema instalado num computador com arquitectura paralela que controla a célula de pintura de uma fábrica de camiões [Morley-1993]. Há 7 estufas de pintura, menos que as cores possíveis, e a mudança de côr numa estufa leva um tempo considerável de activação (setup) durante o qual a estufa não pode ser usada, para além de algum desperdício de tinta. Contudo, se não se efectuarem suficientes trocas de côr haverá encomendas que não cumprirão os seus prazos de entrega. Estabelece-se um protocolo de negociação que atribui carrocerias de camião às estufas. Este sistema levou a poupanças consideráveis na empresa.

A LMS (Logistics Management System) é uma empresa de semicondutores do grupo IBM [Fordyce-1992]. Há uma parte do sistema que usa um protocolo de votação entre 4 agentes que representam “assessores” com responsabilidade de garantir um objectivo cada:
- completar cada lote tão próximo quanto possível dos prazos de entrega;
- atingir quotas de produção diárias;
- satisfação de pedidos para centros subaproveitados;
- reduzir tempos de setup e incrementar a utilização das máquinas.

Desse modo consegue-se uma função global de avaliação que considere vários critérios.

O ADS da Hitachi é uma arquitectura guiada por dados (data driven) para controlo em tempo real que suporta Agentes heterogéneos. O ADS foi usado pela Kawasaki Steel. As mensagens não são dirigidas, mas postas num “data field” que contém um código de conteúdo de modo que os agentes saibam se lhes são destinadas [Mori-1988].

O sistema AARIA foi construído de modo a demonstrar que juntando um grupo de agentes represerntando habilidades de produção é possível criar uma empresa de produção cujo desempenho e funcionalidades suplantem a dos sistemas convencionais [Parunak-1997]. O sistema tem funcionalidades ERP (Enterprise Resouce Planning) e MES (Manufacturing Execution Planning), nomeadamente na recepção de encomendas, compras, gestão de inventários, gestão de recursos, gestão de pessoal, contabilidade, escalonamento de capacidade finita e simulação.

No sistema AARIA existem agentes separados (com igual grau de inteligência e responsabilidade) que representam Partes (componentes), Recursos e Processos Unitários (operações), não havendo um controlo centralizado. As partes movem-se de processo unitário em processo unitário, através de buffers. Cada processo unitário tem como entrada um ou mais componentes (vindos de um ou mais buffers) e produz como saída um ou mais componentes (colocando-os num ou mais buffers). Cada agente efectua um escalonamento local e a política de escalonamento é baseada em janelas temporais dinâmicas, sendo estabelecida com base num acordo com o cliente e o fornecedor com base numa função que considera o custo e a data de entrega. Este sistema foi ainda estendido, de modo a cobrir a cadeia de fornecimento, ligando clientes, fornecedores e instalações fabris distintas.

O HMS Testbed foi desenvolvido na Universidade Católica de Leuven, Bélgica, consistindo num protótipo de uma estação de montagem holónica. Os recursos físicos do sistema envolvem várias estações de trabalho para maquinação e montagem, bem como um sistema de transporte (todos representados por holons). Além desses holons (hardware + software) existem três outros holons (apenas software) para escalonamento, planeamento e controlo.

Neste trabalho o escalonamento não é elaborado por vários holons, mas por um só, sendo o objectivo do trabalho analisar a cooperação entre o elemento de controlo e o de escalonamento. O holon de controlo assume a escala de tarefas enviada pelo holon de escalonamento como sendo uma proposta, tentando cumprir tal proposta se isso for viável. Em situações não previstas o holon de controlo decide autonomamente o que fazer, enviando também um pedido para o holon de escalonamento no sentido de que este gere uma nova escala [Valckenaers-1994].

13.3 – Controlo de Tráfego

O controlo de trafego aéreo é uma aplicação crítica caracterizada pela complexidade, mas que tem uma natureza claramente distribuída.

Podemos imaginar Agentes que representem vôos, filas de aviões, pistas, etc. Também podemos ter Agentes que considerem aspectos como, por exemplo, os factores climatéricos.

O problema é claramente um problema de escalonamento e até podemos fazer uma comparação com um sistema industrial:

Vôo ⇔ Ordem de Fabrico, Encomenda;
Pista ⇔ Máquina;
Tempos entre aviões na pista ⇔ Tempos de Activação (Setup).

O OASIS é um Sistema Multi-Agente implementado usando a plataforma dMARS da AAII e que foi usado com dados do Aeroporto Internacional de Sidney, na Austrália. O OASIS mistura agentes que representam funções (coordenador, sequenciador, definidor de trajectórias, analisador de ventos, interface) com agentes gerados dinamicamente e que representam os aviões que se apresentam para pousar ou levantar vôo.

Outro exemplo relacionado com o tráfego é a arquitectura ADS da Hitachi, já citada anteriormente, que já foi usa para o sistema de controlo de tráfego dos comboios Shinkansen no Japão.

Outro exemplo típico de controlo de tráfego corresponde à coordenação entre semáforos numa cidade. O sistema DVMT usa uma rede de sensores distribuídos espacialmente para o controlo de semáforos, obtendo-se vantagens relativamente a um controlo centralizado.

13.4 – Filtragem de Informação

As aplicações são muito variadas, indo desde os sistemas de processamento inteligente de alarmes que se encontram com frequência em sistemas industriais até a filtragem de informação recebida por um utilizador (por exemplo, via email).

No primeiro caso estamos geralmente na presença de sistemas críticos que se caracterizam por gerarem uma grande quantidade de informação nas situações mais problemáticas (por exemplo, no caso de avaria), a maioria dessa informação corresponde a "side-effects" (efeitos colaterais), pouco importantes para a classificação ou diagnóstico do problema, mas que acaba por complicar muito a tarefa de um operador que tente compreender em tempo útil aquilo que ocorreu. Aqui o Agente é visto como um assistente do operador (designado, muitas vezes como um Sistema de Apoio à Decisão), e em certos casos há a confiança suficiente para passar o controlo para o Agente de modo a que este possa tratar da resolução do problema.

O Sistema Pericial SPARSE é um caso típico de um sistema inteligente que se encontra em evolução para um Agente desse tipo [Vale-1997].

Hoje podemos claramente ver o binómio Internet/WWW como a maior fonte de informação que alguém pode aceder. Muitos dos utilizadores dessa enorme fonte de informação debatem-se com o problema de serem claramente inundados com informação proveniente de diversos meios de acesso ou colecção de informação (mensagens recebidas via correio electrónico, pesquisas usando um browser, etc). É exactamente para facilitar a vida destes utilizadores que faz todo o sentido o uso de Agentes que efectuem a filtragem de informação.

O MAXIMS é um Agente de filtragem de mensagens de correio electrónico que usa aprendizagem automática sobre as actividades que o utilizador efectua sobre as mensagens (apagar, re-enviar, arquivar, responder, etc), efectuando uma aprendizagem por observação do utilizador. Por exemplo, o sistema pode concluir que o utilizador apaga sempre as mensagens de outro utilizador específico sem sequer as ler.

O NEWT é um sistema que aconselha um dado utilizador sobre quais os artigos que deve ler. O sistema baseia-se num processo de exemplos de treino, no qual o utilizador indica artigos que leria e artigos que não leria. Além disso o utilizador pode dar ordens precisas, do género "forneça todos os artigos que tenham no título o termo agente".

13.5 – Agentes de Interface e de Obtenção e Classificação de Informação

A visão de um Agente como elemento de interface com o utilizador que visa facilitar a execução de tarefas de obtenção e de classificação de informação parece uma das aplicações com mais potencial dos agentes, nomeadamente porque tais tarefas são, muitas vezes, desempenhadas de um modo repetitivo e enfadonho.

A obtenção de informação (information gathering) distingue-se da filtragem de informação pelo facto de existir um objectivo mais claro e do Agente providenciar pela procura da informação requerida, mesmo que para tal tenha que se socorrer de diversas fontes.

Hoje em dia grande parte dos sistemas de obtenção de informação baseiam-se na procura de palavras-chave que caracterizem os documentos. Relativamente a escrita de documentos na WWW tem sido reconhecida a necessidade de dar uma maior ênfase ao conteúdo (XML) que ao aspecto (HTML). Estão em maturação muitas técnicas que permitirão em breve uma melhor compreensão dos textos com base em técnicas de Língua Natural e Text Mining.

A classificação de informação pode ter como base a observação do comportamento de um utilizador na execução de uma tarefa de classificação, agindo posteriormente o sistema por imitação. Contudo, a classificação de informação pode requerer técnicas elaboradas de reconhecimento de padrões, usuais em reconhecimento de voz e de imagem.

De seguida iremos descrever alguns sistemas de interface com o utilizador para obtenção e classificação de informação.

O GALAXY é um sistema distribuído e descentralizado com uma interface por voz que permite o acesso a informação sobre um vasto leque de domínios. Este sistema permite, por exemplo, que o utilizador possa obter informação sobre voos para um dado destino, saber quais as condições climatéricas nesse destino, quais os hotéis, como chegar a um dado ponto da cidade, etc. As fontes de conhecimento podem ser várias e como se imagina a Internet serve de hospedeira ideal para este tipo de sistemas [Zue-1995].

A ZDL (Zuno Digital Library) é uma biblioteca digital que fornece uma colecção de dados organizados e serviços para o utilizador fazer uso desses dados. É um sistema multi-agente que fornece ao utilizador uma vista coerente de conjuntos de dados que podem estar desorganizados ou incoerentes (como na WWW). Os agentes podem estar classificados nas seguintes classes:

- Consumidores de informação
- Fornecedores de informação
- Facilitadores que estabelecem a ligação entre os fornecedores e consumidores.

O ARIA é um agente de interface que ajuda o utilizador na tarefa de anotação e acesso a imagens [Lieberman-2001]. Foi desenvolvido em conjunto pelo Massachusetts Institute of Technology e pelos laboratórios da KODAK. Não visa a substituição do utilizador, mas a simplificação e auxílio nas tarefas que este tem de executar.

13.6 – Aplicações Aeroespaciais

Em 1998 foi lançado o DS-1 (Deep Space One) a primeira missão do novo programa da NASA, designado NMP (New Millennium Project). O DS-1 teve como missão a aproximação ao asteróide McAuliffe, a Marte e ao cometa West-Kohoutek-Ikemura. Esta nave, com apenas 100 quilos, foi a primeira a usar o sistema de propulsão iónica. Para além dessa tecnologia, o DS-1 testou 12 outras novas tecnologias, incluindo a operação e navegação autónoma baseada em Inteligência Artificial e tecnologia de agentes. Para o efeito foi desenvolvido o Remote Agent e o AutoNav. O Remote Agent consiste num conjunto de 3 módulos independentes para planeamento, execução de planos e monitorização, estando implementado em LISP sobre um processador RS6000 especial. O AutoNav é o sistema de navegação óptico, usando uma arquitectura idêntica a de um sistema pericial.

Com base no sucesso alcançado na navegação autónoma com o Remote Agent, a NASA lançou o CASPER (Continuous Activity Scheduling, Planning Execution, and Replanning) para o comando de uma missão de 3 sátelites que irá decorrer em 2002. Serão controlados parâmetros de navegação dos satélites e tomadas decisões sobre quais as imagens a tomar e quais deverão ser enviadas para terra.

13.7 – Aplicações no Mercado da Energia Eléctrica

As ferramentas utilizadas e decisões tomadas pelas empresas do sector num ambiente competitivo dependem da estrutura e regras do mercado. Em qualquer tipo de mercado o objectivo é maximizar o lucro. As regras de operação devem ser previamente definidas por entidades independentes, para serem completas e "justas". Justas, neste caso, para não haver conspiração, para a informação no mercado ser aberta a todos, para o acesso à transmissão e distribuição não ser discriminatório, e para que os preços sejam apropriados.

Alterar os regulamentos afecta todas as empresas e o seu modo de negociar. Para se manterem competitivas são necessárias novas ferramentas capazes de ajudar as empresas a transitarem do ambiente antigo para o novo e competitivo mundo do futuro.

Em [Sheblé-99] é possível encontrar vários métodos e ferramentas capazes de serem úteis na indústria eléctrica competitiva.

Actualmente os governos encorajam a abertura do mercado, para criar um ambiente competitivo em que a geração e serviços de suporte são comprados e vendidos face à procura do mercado. O mercado irá consistir em Produtores (PROD), Distribuidores (DIST), Transmissores (TRANS), uma unidade central coordenadora para promover o funcionamento independente do sistema (OIS), e negociadores entre compradores e vendedores (NEG). Os serviços que suportam a entrega fiável de energia, tendo em conta aspectos como as perdas de transmissão, potência reactiva, gestão de congestionamentos, entre outros, têm também que

ser considerados como parte integrante do sistema (AUX). As diversas entidades referidas podem ser vistas como agentes (figura 18).

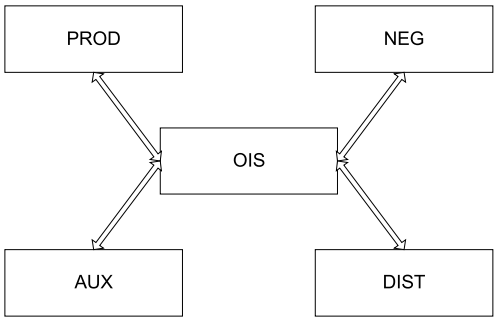

Fig. 18 – Nova Estrutura Organizacional do Mercado da Energia Eléctrica

O Operador Independente do Sistema (OIS) é independente das outras entidades, e embora o seu papel ainda não esteja completamente definido, é responsável por coordenar os participantes no mercado de modo a promover o funcionamento fiável do sistema. O OIS necessita de novos algoritmos de optimização para funcionamento baseado no preço. Em [Sheblé-99] propõe-se um modelo de negociação de energia, utilizado no desenvolvimento de ferramentas de análise e simulação para estudar aspectos de implementação de vários contratos num mercado aberto.

Em [Praça-2001b] é feita a proposta de modelar o mercado da Energia Eléctrica através de um Sistema Multi-Agente. Tal sistema tem como finalidade efectuar a simulação de tal mercado e a análise dos vários cenários possíveis é feita com recurso à Teoria de Jogos, os intervenientes no mercado são vistos como potenciais jogadores e a aplicação de um método similar ao MINIMAX serve para que se definam quais os cenários que interessa analisar de modo a que se possa suportar alguém (comprador ou vendedor) na tomada de decisões.

Referências Bibliográficas

[Agorics-URL] Agorics, Inc., "Auctions", 1996, http://www.agorics.com/new.html.

[Alho-1998] K. Alho (1998) *A Comparison of CORBA, DCOM and RMI*. Helsinki University of Technology HeCSE Winter School. Janeiro de 1998. http://wwwseg.cs.hut.fi/~kta/corba-comparison/

[Bratman-1987] M. Bratman, D. Israel, M. Pollack; Toward an architecture for resource-bounded agents; Technical Report CSLI-87-104; Center of the study of Language and Information; SRI and Stanford University; 1987

[Breiter-1996] P. Breiter, M. Sadek; A Rational Agent as a Kernel of a Cooperative Dialogue Systems: Implementing a Logic Theory of Interaction; Proceedings of ECAI'96 Workshop on Agent Theories, Architectures and Languages; Springer-Verlag, Berlin, pp. 261-276; 1996

[Brooks-1985] R. Brooks; A robust layered control system for a mobile robot; Technical Report AIM-864, MIT AI Lab., Cambridge MA, 1985

[Brustolini-1991] J. Brustolini; Autonomous Agents: characterization and requirements; Carnegie Mellon Technical Report CMU-CS-91-204; Pittsburgh; 1991

[Chavez-1996] A. Chavez, P. Maes; Kasbah: An Agent Marketplace for Buying and Selling goods; Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, Londres (Reino-Unido), 1996

[Chung-1997] P. E. Chung, Y. Huang, S. Yajnik, D. Liang, J. C. Shih, C.-Y. Wang, and Y. M. Wang (1997) *DCOM and CORBA Side by Side, Step By Step, and Layer by Layer*. C++ Report Magazine, Setembro 1997.

[Coelho-1994] H. Coelho; Inteligência Artificial em 25 lições; Fundação Calouste Gulbenkian; 1994

[Collins-1998] J. Collins, B. Youngdahl, S. Jamison, B. Mobasher, M. Gini; A market architecture for Multi-Agent contracting; 2nd International Conference on Autonomous Agents; Minneapolis, Estados Unidos da América; 1998

[Davis-1980] R. Davis; Report on the Workshop on Distributed Artificial Intelligence; SIGART Newsletter, n. 73, pp. 42-52; October 1980

[Davis-1983] R. Davis, R. Smith; Negotiation as a metaphor for Distributed Problem Solving; Artificial Intelligence, vol. 20, n. 1, pp. 63-109; 1983

[Engelmore, 1988] R. Englemore, T. Morgan; Blackboard Systems; Addison-Wesley.

[Fordyce-1992] K. Fordyce, R. Dunki-Jacobs, B. Gerald, R. Sell, G. Sullivan; Logistics Management System: an advanced Decision Support System for the

Fourth Decision Tier Dispatch on Short-Interval Schedulling; Production and Operations Management; vol. 1, n. 1, pp. 70-86; 1992

[Franklin-1996] S. Franklin, A. Graesser; Is it an Agent or just a Program?: A Taxonomy for Autonomous Agents; Third International Workshop on Agent Theories, Architectures and Languages; Springer-Verlag; 1996

[Geist-1994a] A. Geist, A. Beguelim, J. Dongarra, W. Jiang, R. Mancheck, PVM 3 User's Guide and Reference Manual, Technical Report ORNL/TM - 12187, Oak Ridge National Laboratory, Maio 1994.

[Geist-1994b] A. Geist, PVM: A User's Guide and Tutorial for Networked Parallel Computing, MIT Press 1994. http://www.netlib.org/pvm3/book/pvm-book.html

[Gray-1995] Gray, Agent Tcl: Alpha release 1.1, Computer Science Department, Dartmouth College, USA

[Gruber-1993] T. Gruber; A translation approach to portable ontologies; Knowledge Acquisition; vol. 5, n. 2, pp 199-220; 1993

[Guttman-1998] R. Guttman, P. Maes; Agent mediated integrative negotiation for Retail Electronic Commerce; Proceedings of the Workshop on Agent Mediated Electronic Trading, Minneapolis, Estados Unidos da América, 1998

[Hayes-Roth-1995] B. Hayes-Roth; An Architecture for Adaptive Intelligent Systems; Artificial Intelligence; Special Issue on Agents and Interactivity, 72, pp. 329-365; 1995

[Koestler-1967] A. Koestler; The Ghost in the Machine; Hutchinson & Co. Ltd., 1967

[Labrou-1988] Y. Labrou, . Finin; Semantics and Conversations for an Agent Communication Language; em Readings in Agents; M. Huhns, M. Singh; Morgan Kaufmann, San Mateo CA, pp. 235-242; 1988

[Lieberman-2001] H. Lieberman, E. Rosenzweig, P. Singh; Aria: An Agent for annotating and retrieving images; IEEE Computer, vol. 34, n. 7, pp. 57-62; July 2001

[MPI-1995] MPI: A Message-Passing Interface Standard. Message Passing Interface Forum Junho 1995.

[Maes-1995] P. Maes; Artificial Life meets Entertainment: life like Autonomous Agents; Communications of the ACM, 38, 11, 108-114; 1995

[META-1998] META Group Consulting (1998) *CORBA vs. DCOM - Solutions for the Enterprise*; Março de 1998;

http://www.sun.com/swdevelopment/news/CORBA.shtml

[Minsky-1986] M. Minsky; The Society of Mind; Simon and Schuster;1986

[Mitsubishi-1997] Mitsubishi Electric ITA, Concordia: an infrastructure for collaborating Mobile Agents, First International Workshop on Mobile Agents, 1997

[Mori-1988] J. Mori, H. Torikoshi, K. Nakai, K. Mori, T. Masuda; Computer Control System for Iron and Steel Plants; Hitashi Review; vol. 37, n. 4, 251-8; 1988

[Morley-1993] R. Morley, C. Schelberg; An analysis of Plant-specific Dynamic Scheduller; NSF Workshop on Dynamic Schedulling; 1993

[Nilsson-1981] N. Nilsson; Distributed Artificial Intelligence; Report SRI International; Menlo Park CA; 1981

[Nwana-1996] H. Nwana; Software Agents: an overview; Knowledge Engineering Review; vol. 11, n. 3, pp. 205-244; 1996

[Odyssey] Odyssey at General Magic Inc., http://www.genmagic.com/agents/

[Praça-2001a] I. Praça, M.J. Viamonte; Mecanismos de Negociação em Leilões; Relatório Interno, ISEP; 2001

[Praça-2001b] I. Praça, C. Ramos, Z. Vale, Modelling and Simulation of Electricity Markets: Game Theory to improve Decision Support, 15th European Simulation Multiconference, pp.419-421, Praga (República Checa), 2001

[Parunak-1987] H. Parunak; Manufacturing experience with Contract Net; em Distributed Artificial Intelligence; M. Huns; Pitman; pp. 285-310; 1987

[Parunak-1997] H. Parunak, A. Baker, S. Clark; The AARIA Agent Architecture; Proceedings of the 1st International Conference on Autonomous Agents; Marina del Rey, CA, 1997

[Ramos-1993] C. Ramos; Planeamento e Execução Inteligente de Tarefas em Robótica de Montagem e de Manipulação; Tese de Doutoramento; Faculdade de Engenharia da Universidade do Porto; 1993

[Ramos-1996] C. Ramos, A Holonic Approach for Task Scheduling in Manufacturing Systems, IEEE International Conference on Robotics and Automation, pp. 2511-2516, Minneapolis (Estados Unidos da América), 1996

[Ramos-2001] C. Ramos; Sistemas de Apoio à Decisão com Inteligência Escalável; Lição apresentada no âmbito das Provas de Agregação; DEEC – Faculdade de Engenharia da Universidade do Porto; 2001

[Rao-1991] A. Rao, M. Georgeff; Modeling rational agents within a BDI-architecture; Technical Report 14; Australian AI Institute; Carlton, Australia, 1991

[Rodrígues-1998] J. Rodrígues-Aguilar, F. Martín, P. Noriega, P. Garcia, C. Sierra; Towards a testbed for Trading Agents in Electronic Auction Markets; AI Communications; IOS Press; 1998

[Russell-1995] S. Russell, P. Norvig; Artificial Intelligence a modern approach; Prentice-Hall International Editions, 1995

[Sheblé-1999] G. Sheblé, "Computational Auction Mechanisms for Reestructured Power Industry Operation", Kluwer Academic Publishers, Londres, 1999

[Silva-1998] N. Silva; Sistemas Holónicos de Produção; Tese de Mestrado; Faculdade de Engenharia da Universidade do Porto; 1998

[Smith-1994] D. Smith, A. Cypher, J. Spohrer; KidSim: Programming Agents without a Programming Language; Communications of the ACM, 37, 7, 55-67; 1994

[Sousa-1999] P. Sousa, J. Morais, Análise de Tecnologias de Distribuição, relatório técnico, ISEP, 1999

[Sousa-2000] P. Sousa; Agentes Inteligentes em Sistemas Holónicos de Produção; Tese de Doutoramento; Universidade do Minho; 2000

[Valckenaers-1994] P. Valckenaers, F. Bonneville, H. Van Brussel, L. Bongaerts, J. Wyns; Results of the Holonic Control System Benchmark at the University of Leuven; Proceedings of the International Conference on Computer Integrated Manufacturing and Automation Technology; pp. 128-133, RPI, Troy, NY, Estados Unidos da América, 1994

[Vale-1997] Z. Vale, A. Moura, M. Fernandes, A. Marques, C. Rosado, C. Ramos; Sparse: An Intelligent Alarm Processor and Operator Assistant; IEEE Expert - Intelligent Systems and Applications, vol. 12, n. 3, pp. 86-93, May/June 1997

[Vetter-2000] M. Vetter, S. Pitsch; Towards a Flexible Trading Process over the Internet; The European AgentLink Perspective, Lecture Notes in Artificial Intelligence 1991, Frank Dignum, Carlos Sierra, Springer, pp. 148-162, 2000

[Viamonte-2000] M.J. Viamonte, C. Ramos, A model for an Electronic Market Place, em Agent Mediated Electronic Commerce – The European AgentLink Perspective, Lecture Notes in Artificial Intelligence 1991, Frank Dignum, Carlos Sierra, Springer, pp. 115-125, 2000

[Walker-1994] J. Walker, L. Sproull, R. Subramani; Using a human face in an interface; Proceedings of Computer-Human Interaction; pp. 85-91; Boston (Estados Unidos da América); 1994

[Wittig-1992] T. Wittig; ARCHON – An architecture for Multi-Agent Systems; Ellis Horwood Ltd; 1992

[Wooldridge-1995] M. Wooldridge, N. Jennings; Agent Theories, Architectures, and Languages: a Survey; Intelligent Agents; M. Wooldridge, N. Jennings; Springer-Verlag, 1-22; 1995

[Wooldridge-1998] M. Wooldridge, N. Jennings; Pitfalls of Agent-oriented Development; 1998

[Wooldridge, 2002] M. Wooldrigde; Introduction to MultiAgent Systems, Addison-Wesley, 2002

[Wurman-1998] P. Wurman, M. Wellman, W. Walsh, The Michigan Internet AuctionBot: A configurable auction server for human and software agents; 2nd International Conference on Autonomous Agents; Minneapolis, Estados Unidos da América; 1998

[Zue-1995] Zue V.; Navigating the Information Superhighway using Spoken Language Interfaces; IEEE Expert Intelligent Systems and their Applications, vol. 10, n. 5, pp. 39-43, October 1995