APPLICATIONS OF MULTI-AGENT SYSTEMS

Mihaela Oprea
University of Ploiesti, Department of Informatics, Bd. Bucuresti Nr. 39, Ploiesti, Romania

Abstract: Agent-based computing has the potential to improve the theory and practice of modelling, designing, and implementing complex systems. The paper presents that basic notions of intelligent agents and multi-agent systems and focus on some applications of multi-agent systems in different domains.

Key words: intelligent agents; multi-agent systems; coordination; negotiation; learning; agent-oriented methodologies; applications.

1. INTRODUCTION

In the last decade, Intelligent Agents and more recently, Multi-Agent Systems appeared as new software technologies that integrate a variety of Artificial Intelligence techniques from different subfields (reasoning, knowledge representation, machine learning, planning, coordination, communication and so on), and which offer an efficient and more natural alternative to build intelligent systems, thus giving a solution to the current complex real world problems that need to be solved. For example, a complex system could be decomposed in components and again the components in sub-components and so on, till some primitive entities are obtained. Some of these primitive entities could be viewed as being agents that solve their local problems and interact between them in order to solve the goal of the initial complex system. However, most of the real world complex systems are only nearly decomposable, and a solution would be to endow the components with the ability of making decisions about the nature and the scope of their interactions at run time. Still, from this simplistic view, we could figure out a new type of computing, based on agents. In [1], Jennings argued that agent-

of the Fishmarket system is that it is left to the buyers and sellers to encode their own bidding strategies. Also, the auctions could be monitored by the FM Monitoring Agent that keeps track of every single event taking place during a tournament. In Fishmarket each agent in a MAS is dynamically attached to a controller module, and it's in charge of controlling its external actions (i.e. protocol execution).

3.11 SARDINE

In [62] it is described an alternative airline flight bidding prototype system, called SARDINE (System for Airline Reservations Demonstrating the Integration of Negotiation and Evaluation), which offers better choices in comparison to the Priceline system. The SARDINE system uses software agents to coordinate the preferences and interests of each party involved. The buyer agent takes the buyer's preferences and correlates these parameters with the available flights from a reservation database. The user then tells the buyer agent how much to bid and the airline agents accept the ticket bids from the buyer agent. Finally, the airline agents consider individual bids based on flight yield management techniques and specific buyer information. The SARDINE system uses the OR combinatorial auction. A combinatorial auction is one in which the user submits simultaneous multiple bids. The bids are mutually exclusive of one another, thus an OR combinatorial auction is used.

3.12 eMediator

The eMediator [63], [64] is a next generation electronic commerce server that has three components: an auction house *(eAuctionHouse),* a leveled commitment contract optimizer *(eCommitter),* and a safe exchange planner *(eExchangeHouse).* The eAuction House allows users from Internet to buy and sell goods as well as to set up auctions. It is a third party site, and therefore both sellers and buyers can trust that it executes the auction protocols as stated. It is implemented in Java and uses some of the computationally intensive matching algorithms in C++. In order to increase reliability the information about the auctions is stored in a relational database. The server is the first Internet auction that supports combinatorial auctions, bidding via graphically drawn price-quantity graphs, and by mobile agents.

oriented approaches can significantly enhance our ability to model, design and build complex (distributed) software systems.

A natural way to modularise a complex system is in terms of multiple, interacting autonomous components that have particular goals to achieve, i.e. of a multi-agent system (MAS). A multi-agent approach is an attempt to solve problems that are inherently (physically or geographically) distributed where independent processes can be clearly distinguished. Such problems include, for example, decision support systems, networked or distributed control systems, air traffic control. Therefore, multi-agent systems approach is appropriate for distributed intelligence applications: network based, human involved, physically distributed, decentralized controlled, etc.

The basic notion of agent-computing is the agent with its derivation, the software agent. Several definitions has been given to the notion of agent. According to Michael Wooldridge, an *agent* is a computer system that is situated in some environment, and is capable of flexible, autonomous action in that environment in order to meet its design objectives [2]. The flexibility characteristic means that the agent is reactive, pro-active and social. Therefore, the key characteristics of agents are autonomy, proactivity, situatedness, and interactivity. More characteristics could be added, such as mobility, locality, openness, believability, learning, adaptation capabilities, comprehensibility, etc. A *software agent* is an independently executing program able to handle autonomously the selections of actions when expected or limited unexpected events occur.

Summarizing, an agent need to have computational abilities (reasoning, searching, etc) and can use its knowledge and rationality models to map inputs to outputs that would maximize its utility (its performance measure according to the rationality). According to the interaction strategy that is used, an agent could be *cooperative, self-interested,* and *hostile*. Cooperative agents could work together with other agents and humans with the intention of solving a joint problem. Self-interested agents try to maximize their own goods without any concern for the global good, and will perform services for other agents only for compensation (e.g. in terms of money). Hostile agents have a utility that increases with their own gains, and increases also with the competitor's losses.

The agents can be viewed as living in a society where they have to respect the rules of that society. They also live in an organization, which can be effectively executed only in respect with organizational patterns of interactions. In general, multi-agent systems represent institutions where agents must fulfill a set of expected behaviours in their interactions.

2. MULTI-AGENT SYSTEMS

Multi-agent systems are a particular type of distributed intelligent systems in which autonomous agents inhabit a world with no global control or globally consistent knowledge. Figure 1 presents the so called multi-agent system equation, which states that in a multi-agent system a task is solved by agents that communicate among them.

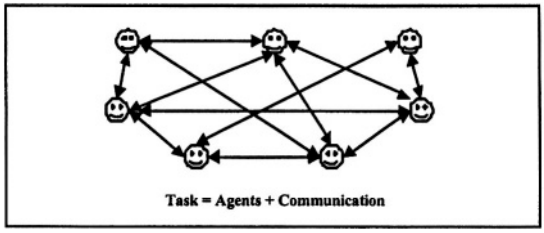

Figure 1. The multi-agent system equation.

We could view a multi-agent system as a society of individuals (agents) that interact by exchanging knowledge and by negotiating with each other in order to achieve either their own interest or some global goal. One of the characteristics of some MASs is the openness, which means that new agents can be created or can enter a MAS (i.e. mobile agents can arrive), and some unknown entities (e.g. legacy and elsewhere implemented entities) may enter a MAS. This characteristic has some technological implications: the need for standards (such as FIPA [3]) and the existence of a proper infrastructure that support interoperations.

In a MAS, agents are embedded in a certain environment which could be dynamic, unpredictable and open. This environment is the world of resources the agents perceive. The interactions between agents are the core of a multi-agent system functioning. Starting from [4], Nick Jennings has introduced the definition of a new computer level, the Social Level (SL) [5] in order to solve the problems related to flexible social interactions. With a SL incorporated, above the Knowledge Level (KL), the prediction of the behaviour of the social agents and of the whole MAS could be easily made. Following the Newell's notation, a preliminary version of the SL is given by: the system (an agent organization), the components (primitive elements from which the agent organization is built), composition laws (e.g. roles of agents in the organization), behaviour laws, and the medium (the elements the system processes in order to obtain the desired behaviour). The social level allows to create organizational models of multi-agent systems.

In a multi-agent system, agents are connected through different schemes, usually following mesh and hierachical structures.

The main characteristics of a multi-agent system are: *autonomy* (agents may be active and are responsible for their own activities), *complexity* (induced by the mechanisms of decision-making, learning, reasoning, etc), *adaptability* (adjust the agents activities to the dynamic environmental changes), *concurrency* (in case of tasks parallel processing), *communication* (inter-agent, intra-agent), *distribution* (MASs often operate on different hosts and are distributed over a network), *mobility* (agents need to migrate between platforms and environments), *security and privacy* (possible intrusion of the agents' data, state, or activities), and *openness* (MASs can dynamically decide upon their participants).

A multi-agent system has functional and non-functional properties. The functional properties are coordination, rationality, knowledge modelling. The non-functional properties are performance (response time, number of concurrent agents/task, computational time, communication overhead, etc), scalability (increased loading on an agent which is caused by its need to interact with more agents because the size of the society has increased), stability (a property of an equilibrium). The non-functional properties are discussed in [6]. Scalability is a property that becomes important when developing practical MASs. Most agent systems that have beein built so far involve a relatively small number of agents. When multi-agent systems are employed in larger applications this property need a very careful analysis. The scalability of a MAS is the average measure of the degree of performance degradation of individual agents in the society as their environmental loading increases, due to an expansion in the size of the society [6].

In a multi-agent system, agents have only local views, goals and knowledge that may conflict with others, and they can interact and work with other agents for the desired overall system's behavior. In order to achieve the common goals, a multi-agent system need to be coordinated. Coordination has to solve several problems such as the distributed expertise, resources or information, dependencies between agents' actions, efficiency. Two main dependencies can be encountered in a MAS, inter-agent dependencies and intra-agent dependencies. Several approaches that tackle the coordination problem were developed in Distributed Artificial Intelligence and Social Sciences, starting with diferent interaction protocol, partial global planning, and ending with the social laws. Depending on the domain of application specific coordination techniques are more appropriate. Also, the type of coordination protocol that is employed will influence the performance of the MAS.

A multi-agent infrastructure has to enable and rule interactions. It is the "middleware" layer that supports communication and coordination activities. The communication infrastructures (e.g. FIPA defined communication infrastructures) are dedicated to the control of the global interaction in a MAS. It includes routing messages and facilitators. The coordination infrastructures (e.g. MARS and Tucson coordination infrastructures [7]) are dedicated to laws that establish which agents can execute which protocols and where. It includes synchronization and constraints on interactions.

The main benefits of multi-agent systems approaches are the following: address problems that are too large for a centralized single agent (e.g. because of resource limitations or for robustness concerns), allow the interconnection and interoperation of multiple existing legacy systems (e.g. expert systems, decision support systems, legacy network protocols), improve scalability, provide solutions to inherently distributed problems (e.g. telecommunication control, workflow management), and provide solutions where the expertise is distributed. Some of the problems that could appear in a MAS are related to the emergent behaviour, system robustness, and system reliability.

A major characteristic in agent research and applications is the high heterogeneity of the field. The heterogeneity means agent model heterogeneity (different models and formalisms for agents), language heterogeneity (different communication and interaction schemes used by agents), and application heterogeneity (various goals of a MAS for many application domains). The heterogeneity has to be manageable with appropriate models and software toolkits. In the next sections we briefly discuss some models for agent architectures, communication, coordination, negotiation, learning in a multi-agent system, and, finally, we made a short presentation of the most known and used MAS development methodologies, and MAS development software.

2.1 Agent architectures

Two main complementary approaches are currently used to characterize intelligent (i.e. rational) agents and multi-agent systems: operational (agents and MASs are systems with particular features, i.e. a particular structure and a particular behaviour), and based on system levels (agents and MASs are new system levels). The first approach define rational agents in terms of *beliefs* (information about the current world state), *desires* (preferences over future world states) and *intentions* (set of goals the agent is committed to achieve) - BDI, thus being independent from the internal agent architecture. The advantage is that uses the well founded logics (e.g. modal logics). One of the problems is related to the ground of rationality on axioms of a logic.

The second approach hides details in hardware design. System levels are levels of abstraction. The agent is modelled as being composed of a body (i.e. means for the agent to interact with its environment), a set of actions the agent can perform on its environment, and a set of goals.

Figure 2 presents the general architecture of an agent.

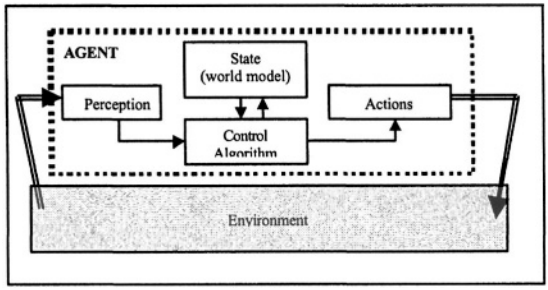

Figure 2. The general architecture of an agent.

The main agent architectures reported in the literature (see e.g. [8]) are the deliberative architecture, the reactive architecture, and the hybrid architecture. Most agent architectures are dedicated to the fulfillment of precise tasks or to problem solving, typically requiring reasoning and planning. Other approaches simulate emotions, which direct the agent in a more reactive way.

An agent that uses *the deliberative architecture* contains a symbolic world model, develops plans and makes decisions in the way proposed by symbolic artificial intelligence. Two important problems need to be solved in this case: the transduction problem and the representation/reasoning problem. The solution of the first problem led to work on vision, speech understanding, learning, etc. The solution for the second problem led to work on knowledge representation, automated reasoning/planning etc. The answer to the question "*how should an agent decide what to do*?" is that it should deduce its best action in light of its current goals and world model, so, it should plan. The world model can be constructed through *learning*.

An agent that uses *the reactive architecture* does not include any kind of central symbolic world model, and does not use complex symbolic reasoning. Rodney Brooks [9] had introduced two ideas: situatedness and embodiment. In his view "intelligent" behaviour arises as a result of an agent's interaction with its environment, and the agent is specified in terms of perceptions and actions. It is not a central dogma of the embodied approach that no internal representations exist. The being of the agent is dependent on the context in which it is encountered, and it is derived from

purposeful interaction with the world. A possible answer to the question "*how should an agent decide what to do*?" is to do planning in advance and compile the result into a set of rapid reactions, or situation-action rules, which are then used for real-time decision making, or to learn a good set of reactions by trial and error.

An agent that uses a *hybrid architecture* has a layered architecture with both components: deliberative and reactive, usually, the reactive one having some kind of precedence over the deliberative one. Two important problems need to be solved: the management of the interactions between different layers and the development of the internal structure of an internally unknown system characterized by its I/O behavior. A possible answer to the question "*how should an agent decide what to do*?" is by integrating planning system, reactive system and learning system into a hybrid architecture, even included into a single algorithm, where each appears as a different facet or different use of that algorithm. This answer was given by the Dyna architecture [10].

How to choose the agent architecture is not an easy problem and is mainly application domain dependent. In [11] it is claimed that evolution has solved this problem in natural systems with a hybrid architecture involving closely integrated concurrently active deliberative and reactive sub-architectures.

2.2 Communication

Interaction is a fundamental characteristic of a multi-agent system. The agent communication capability is needed in most cases for the coordination of activities. A conversation is a pattern of message exchange that two or more agents agree to follow in communicating with one another. Actually, it is a pre-established coordination protocol. Several methods could be used for the representation of conversations: state transition diagrams, finite-state machines, Petri nets, etc. An Agent Communication Language (ACL) is a collection of speech-act-like message types, with agreed-upon semantics, which facilitates the knowledge and information exchange between software agents. The standardization efforts made so far generated a standard framework for agent communication (a standard agent communication language – KQML [12], FIPA ACL – both based on the notion of speech act). Current used ACLs are not accepted by all researchers due to some problems they have: the formal semantics for such languages (define semantics based on mental states, or equate the meaning of a speech act with the set of allowable responses), the relationships between speech acts and various related entities such as conversations, agent mental state. A possible alternative model of agent communication is *Albatross* (Agent language

based on a treatment of social semantics, [13]) that has a commitment-based semantics for speech acts.

KQML (Knowledge Query and Manipulation Language) is a high level message-oriented communication language and protocol for information exchange independent of content syntax and applicable ontology. It is independent of the transport mechanism (TCP/IP, SMTP, etc), independent of the content language (KIF, SQL, Prolog, etc), and independent of the ontology assumed by the content. A KQML message has three layers: content, communication, and message. The syntax of KQML is based on the s-expression used in Lisp (*performative arguments*). The semantics of KQML is provided in terms of pre-conditions, post-conditions, and completion conditions for each performative. FIPA ACL is similar to KQML. The communication primitives are called *communicative acts* (CA). SL is the formal language used to define the semantics of FIPA ACL. It is a multi-modal logic with modal BDI operators, and can represent propositions, objects, and actions. In FIPA ACL, the semantics of each CA are specified as sets of SL formulae that describes the act's feasability pre-conditions and its rational effect.

A message has three aspects: *locution* (how is the message phrased), *illocution* (how is the message meant by the sender or understood by the receiver), and *perlocution* (how does the message influence the receiver's behavior). Figure 3 shows an example of ACL message.

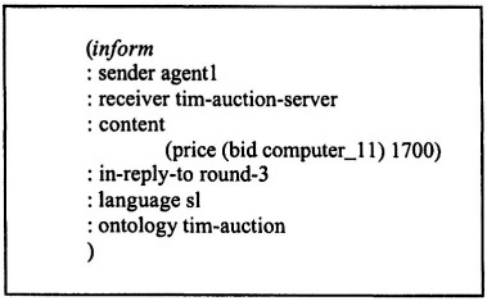

Figure 3. Example of an ACL message.

One important issue in agent communication is the understanding of messages meaning. The message ontology gives the meaning of a message. The ontology provides interpretation of the message, giving a meaning for each word included in the content of the message. More generally, ontology is a description of the concepts and relationships that can exist for an agent. Usually, an ontology is designed for a specific multi-agent system, thus being aplication domain dependent. Therefore, one of the problems that may occur is the communication among agents that use different ontologies.

Agent communication languages such as KQML and FIPA ACL have provided a tool and framework to tackle the interoperability problems of inter-agent communication.

2.3 Coordination

In a multi-agent system agents have their own plans, intentions and knowledge, and are willing to solve their local goals, while for the global goal of the system it is needed a coordination mechanism to solve the conflicts that may arise due to limited resources or to the opposite intentions the agents might have. Coordination is a process in which the agents engage in order to ensure that the multi-agent system acts in a coherent manner. For this purpose, the agents must share information about their activities. One way the agents may achieve coordination is by communication. Another way, without communication, assume that the agents have models of each others' behaviors. Coordination avoids unnecessary activity and allows a reduce resource contention, avoids deadlock, and livelock and maintains safety conditions (minimizing the conflicts). Deadlock refers to a state of affairs in which further action between two or more agents is impossible. Livelock refers to a scenario where agents continuously act (exchange tasks, for example), but no progress is made.

According to [14], any comprehensive coordination technique must have four major components: (1) a set of structures that enable the agents' interaction in predictable ways; (2) flexibility in order to allow the agents to operate in dynamic environment and to cope with their inherently partial and imprecise viewpoint of the community; (3) a set of social structures which describe how agents should behave towards one another when they are engaged in the coordination process; (4) sufficient knowledge and reasoning capabilities must be incorporated in order to exploit both the available structure (individual and social) and the flexibility. In [15], a coordination model is described by three elements: *the coordinates,* i.e. the objects of the coordination (e.g. the software agents), *the coordination media,* i.e. what enables the interaction between the coordinables (e.g. the agent communication language), and *the coordination laws* that govern the interaction between the coordination media and the coordinables, and the rules that the coordination media employs (e.g. the finite state machine that describes the interaction protocol).

Coordination may require cooperation between agents, but sometimes, coordination may occur without cooperation. The design of a specific coordination mechanism will take into account appart from the domain of the application, the type of architecture that is adopted for the MAS design. There are mediated interaction coordination models (e.g. blackboard based

interaction), and non mediated interaction ones. When a mediated interaction coordination protocol is applied, the state of the interaction could be inspected in order to check the coordination trace.

In the literature there have been reported different coordination techniques. In [14] it is made a classification of the existing coordination techniques applied to software agent systems. Four broad categories were identified: organizational structuring, contracting, multi-agent planning, and negotiation. The first category includes coordination techniques like the classical client-server or master-slave techniques. A high level coordination strategy from the second category is given by the Contract Net Protocol (CNP). A multi-agent planning technique, from the third category, involves the construction of a multi-agent plan that details all the agents' future actions and interactions required to achieve their goals, and interleave execution with more planning and re-planning. The fourth category of coordination techniques uses negotiation to solve the conflicts that may arise in a MAS. Usually, negotiation compromises speed for quality because there is an overhead during negotiation before compromise is made.

In [15] it is made a comparison between three types of coordination models: hybrid coordination models based on tuple centers [16], interaction protocols as a coordination mechanism, and implicit coordination through the semantic of classic ACLs. All these models were proposed by different research communities. The coordination community proposed the first one, while the second one was proposed by the agent community, and the last one by the more formally inclined part of the agent community.

In [17] it is presented a framework that enables agents to dynamically select the mechanism they employ in order to coordinate their inter-related activities. The classification made in [17] reveals two extreme classes, the social laws (long-term rules that prescribe how to behave in a given society), and the interaction protocols (e.g. CNP, that coordinates the short-term activities of agents in order to accomplish a specific task). Between these classes it is situated partial global planning. In [18] another approach is described, the use of coordination evaluation signals to guide an agent's behavior. In [19] it is presented a precise conceptual model of coordination as structured "conversations" involving communicative actions amongst agents. The model was extended with the COOL (COOrdination Language) language and it was applied in several industrial MAS. A social based coordination mechanism is described in [20]. Coordination gradually emerges among agents and their social context. The agents are embedded in a social context in which a set of norms is in force. These norms influence the agents' decision-making and goal generation processes by modifying their context.

In cases where the communication costs are prohibitive, a solution is to coordinate a set of individually motivated agents by choosing an equilibrium point. Such an evolutionary approach is used in [21], where by learning from observations the coordination point (viewed as an equilibrium point) is reached.

Another coordination model is given by stigmergy [22], [23], which means that agents put signs, called sigma in Greek, in their environment to mutually influence each other's behaviour. Such indirect coordination mechanism is suitable for small-grained interactions compared to coordination methods that usually require an explicit interaction between the agents. With stigmergy, agents observe signs in their environment and act upon them without needing any synchronization with other agents. The signs are typically multi-dimensional and reflect relevant aspects of the coordination task. For example, the display of apples provides the agents with information through look, smell, and packaging of the apples. A multi-agent coordination and control system design, inspired by the behaviour of social insects such as food foraging ants is discussed in [24].

2.4 Negotiation

Negotiation is a discussion in which interested parties exchange information and, eventually, come to an agreement [25]. In a multi-agent system, negotiation has been viewed as a solution for problems such as network coherency, the problem decomposition and allocation, and more generally for the coordination problems. Negotiation can be viewed as a process whereby agents communicate to reach a common decision. Thus, the negotiation process involves the identification of interactions (through communication) and the modification of requirements through proposals and counter-proposals. The main steps of negotiation are (1) exchange of information; (2) each party evaluates information from its own perspective; (3) final agreement is reached by mutual selection. Two main types of negotiation were reported in the literature [26], [27], [28]: distributive negotiation and integrative negotiation.

2.4.1 Distributive negotiation

The distributive negotiation (win-lose type of negotiation, such as auctions) involves a decision-making process of solving a conflict between two or more parties over a single mutually exclusive goal. In the game theory this is a zero-sum game. Auctions are methods for allocating tasks, goods, resources, etc. The participants are auctioneer and bidders. Example of applications includes delivery tasks among carriers, electricity, stocks,

bandwidth allocation, heating, contracts among construction companies, fine art, selling perishable goods, and so on.

Different types of distributive negotiation are available: the Contract Net Protocol, and auction mechanisms such as first-price sealed-bid, second-price sealed-bid, English auction, Dutch auction, etc.

One of the problems encountered when adopting an auction mechanism is how to incorporate a fair bidding mechanism, i.e. keeping important information secure or isolated from competing agents. Another problem is that the auction protocol may not always converges in finite time. Still, the main advantage of auctions is that in certain domains, because the goods are of uncertain value, dynamic price adjustment often maximize revenue for a seller.

2.4.2 Integrative negotiation

The integrative negotiation (win-win type of negotiation, such as desired retail merchant-customer relationships and interactions) involves a decision-making process of solving a conflict between two or more parties over a multiple interdependent, but non-mutually exclusive goals. In game theory this is a non-zero-sum game. Usually, integrative negotiation deals with multi-attribute utility theory. Negotiation involves determining a contract under certain terms and conditions. An integrative negotiation model is characterized by three key information: the negotiation protocol, the list of issues over which negotiation takes place, and the reasoning model used by the agents (i.e. the negotiation strategy).

Several negotiation models were proposed, service-oriented negotiation models, persuasive argumentation, strategic negotiation, etc.

A service-oriented negotiation model that was used with success in MAS applications is presented in [29]. Suppose that one agent (the client) requires a service to be performed on its behalf by some other agent (the server). The negotiation between the client and the server may be iterative with several rounds of offers and counter-offers that may occur before an agreement is reached or the negotiation process is terminated. The negotiation can ranged over a set of issues (quantitative and qualitative). The sequence of offers and counter-offers in a two-party negotiation is called the negotiation thread. Offers and counter-offers are generated by linear combination of simple functions called tactics. Different weights in the linear combination allow the varying importance of the criteria to be modeled. Tactics generate an offer or a counter-offer, for a single component of the negotiation issue using a single criterion (time, resource, behavior of the other agent, etc). The way in which an agent changes the weights of the different tactics over time is given by the agent negotiation strategy. Different types of tactics were used by the

service-oriented negotiation model, time dependent tactics (including Boulware, and Conceder tactics), resource dependent tactics (dynamic deadline tactics and resource estimation tactics), and behaviour dependent tactics (relative Tit-For-Tat, random absolute Tit-For-Tat, average Tit-For-Tat). In this model, the agent has a representation of its mental state containing information about its beliefs, its knowledge of the environment and any other attitudes like desires, goals, intentions and so on, which are important to the agent.

Let's consider the case of two agents, a seller and a buyer, that are negotiating the price of a specific product. Each agent knows its own reservation price, RP, which is how much the agent is willing to pay or to receive in the deal. A possible deal can be made only if there exists an overlapping zone between the reservation prices of the two agents. The agents don't even know if there is an agreement zone. The ideal rule to find if there is an agreement zone is given in figure 4.

if $RP^{seller} \leq RP^{buyer}$ **then**
  * Possible deal - the zone of agreement exists;
else
  * No deal - the zone of agreement doesn't exist;

where RP^{buyer} is the maximum price the buyer is willing to pay for the product, and RP^{seller} is the minimum price the seller will accept to receive.

Figure 4 Rule for agreement zone search.

The way in which each agent is making a proposal for the price of the product is given by its pricing strategy. Usually, it cannot be forecast if an agreement will be made between two agents that negotiate. The convergence in negotiation is achieved when the scoring value of the received offer is greater than the scoring value of the counter-offer the agent intend to respond with.

The PERSUADER system [30] was developed to model adversial conflict resolution in the domain of labour relations which can be multi-agent, multi-issue, single or repeated negotiation encounters. The system uses both case-based reasoning and multi-attribute utility theory. The negotiation is modeled as an incremental modification of solution parts through proposals and counter-proposals. In this model the agents try to influence the goals and intentions of their opponents through persuasive argumentation. In [31] and [32] more details are given regarding the negotiation by arguing.

In the strategic negotiation model [33], a game-theory based technique, there are no rules which bind the agents to any specific strategy. The agents are not bound to any previous offers that have been made. After an offer is

rejected, an agent whose turn it is to suggest a new offer can decide whether to make the same offer again, or to propose a new one. In this case, the negotiation protocol provides a framework for the negotiation process and specifies the termination condition, but without a limit on the number of the negotiation rounds. It is assumed that the agents can take actions only at certain times in the set T={0, 1, 2, …}, that are determined in advance and are known to the agents. Strategic negotiation is appropriate for dynamic real-world domains such as resource allocation, task distribution, human high pressure crisis negotiation.

2.5 Learning

In a multi-agent system the agents are embedded in the environment where they live, and need to interact with other agents in order to achieve their goals. Usually, they try to adapt to the environment by learning or by an evolutionary process, thus doing an anticipation of the interaction with the other agents. Learning in a multi-agent environment is complicated by the fact that, as other agents learn, the environment effectively changes. When agents are acting and learning simultaneously, their decisions affect and limit what they subsequently learn.

Adaptability and embodiment are two important issues that need to be addressed when designing flexible multi-agent systems [28], [34]. Adaptability allows the generation of a model of the selection process within the system and thus results in internal representations that can indicate future successful interactions. Agents can be seen as having a "body" that is embedded in their work environment, and is adapted to this environment by learning or by an evolutionary process. In the context of a multi-agent system, the two properties, adaptability and embodiment, are tightly related to each other.

The most used learning algorithms that were experimented in the case of multi-agent systems are reinforcement learning (e.g. Q-learning [35]), Bayesian learning, and model-based learning.

2.5.1 Reinforcement learning

The reinforcement learning is a common technique used by adaptive agents in MASs and its basic idea is to revise beliefs and strategies based on the success or failure of observed performance. Q-learning is a particular reinforcement learning algorithm (an incremental reinforcement learning) that works by estimating the values of all state-action pairs. An agent that uses a Q-learning algorithm selects an action based on the action-value function, called the Q-function. $Q_j(s, a) = Q_j(s, a) + \lambda(r_j - Q_j(s, a))$, where λ is

a constant, r_j is the immediate reward received by agent *j* after performing action *a* in state *s*. The Q-function defines the expected sum of the discounted reward attained by executing an action *a* in state *s* and determining the subsequent actions by the current policy π. The Q-function is updated using the agent's experience.

The reinforcement learning techniques have to deal with the exploration-exploitation dilemma. Some experimental comparisons between several explore/exploit strategies are presented in [36] showing the risk of exploration in multi-agent systems. In [35] it is demonstrated that genetic algorithm based classifier systems can be used effectively to achieve near-optimal solutions more quickly than Q-learning, this result revealing the problem of slow convergence that is specific to reinforcement learning techniques.

2.5.2 Bayesian learning

Usually, Bayesian behaviour is considered as the only rational agent's behaviour, i.e. the behaviour that maximizes the utility. Bayesian learning is built on bayesian reasoning which provides a probabilistic approach to inference. The bayesian learning algorithms manipulates probabilities together with observed data. In [37] it is presented a sequential decision making model of negotiation called Bazaar, in which learning is modeled as a Bayesian belief update process. During negotiation, the agents use the Bayesian framework to update knowledge and belief that they have about the other agents and the environment. For example, an agent (buyer/seller) could update his belief about the reservation price of the other agent (seller/buyer) based on his interactions with the seller/buyer and on his domain knowledge. The agent's belief is represented as a set of hypotheses. Each agent tries to model the others in a recursive way during the negotiation process, and any change in the environment, if relevant and perceived by an agent, will have an impact on the agent's subsequent decision making. The experiments showed that greater the zone of agreement, the better the learning agents seize the opportunity.

2.5.3 Model-based learning

In [38] it is described a model-based learning framework that model the interaction between agents by the game-theoretic concept of repeated games. The approach tries to reduce the number of interaction examples needed for adaptation, by investing more computational resources in deeper analysis of past interaction experience. The learning process has two stages: (1) the learning agent infers a model of the other agent based on past interaction and

(2) the learning agent uses the learned model for designing effective interaction strategy for the future. The experimental results presented in [38] showed that a model-based learning agent performed significantly better than a Q-learning agent.

2.5.4 Nested representations

An important aspect that should be taken into account when designing adaptive multi-agent systems is the utility of nested representations that are essential for agents that must cooperate with other agents [7]. In [39] it is introduced a method of learning nested models, in order to decide when an agent should behave strategically and when it should act as a simple price-taker, in an information economy. In general, learning abilities consists in making a choice from among a set of fixed strategies and do not consider the fact that the agents are embedded in an environment (i.e. inhabit communities of learning agents).

2.6 Methodologies for multi-agent system development

The development of multi-agent systems applications has generated the development of an agent-specific software engineering, called Agent-Oriented Software Engineering (AOSE), which defines abstractions (of agents, environment, interaction protocols, context), specific methodologies and tools, and could be applicable to a very wide range of distributed computing applications. The adoption of object-oriented (OO) methodologies from object-oriented software engineering is an option, but some mismatches could appeared, as each methodology may introduce new abstractions (e.g. roles, organisation, responsibility, belief, desire, and intentions).

Usually, the whole life-cycle of system development (analysis, design, implementation, validation) is covered by a methodology. Let's consider the analysis and design steps. During the analysis step, agents are associated with the entities of the analyzed scenarios. Then, roles, responsibilities and capabilities are associated accordingly. Finally, interaction patterns between agents are identified. At the knowledge level, for each agent we need to identify its beliefs, goals, body (i.e. the way it interacts with the environment), and actions. The environment behaviour should be identified. At the social level, the analysis step focus on the analysis of an organization and it is needed to identify the roles in the organization, the organizational relations between roles, the dependency between roles, the interaction channels, the obligations, and the influence mechanisms. At the agent design step, we associate agents with the components used to build the system.

There are two approaches for tackle the Agent-Oriented Methodologies (AOM), the Knowledge Engineering (KE) approach and the Software Engineering (SE) approach. The KE approach provides techniques for agent's knowledge modelling. Example of such tools are DESIRE and MAS-CommonKADS [40]. The SE approach used the OO approach, in which an agent is viewed as an active object. Examples of such tools are AUML, GAIA, ADEPT, MESSAGE/UML, OPM/MAS [41]. AUML [42] is an extension of the standard SE approach, UML (Unified Modeling Language). The FIPA Agent UML (AUML) standard is under development. MASs are sometimes characterized as extensions of object-oriented systems. This overlay simplified view has often troubled system designers as they try to capture the unique features of MASs using OO tools. Therefore, an agent-based unified modeling language (AUML) is being developed [43]. The ZEUS toolkit was developed in parallel with an agent development methodology [44], which is supported by the ZEUS Agent Generator tool. DESIRE provides formal specifications to automatically generate a prototype. It has a design approach more than an analysis approach. Agents are viewed as composed components, and MAS interaction as components interaction. OPM/MAS offers an approach that combines OO and process orientation. GAIA is a top-down design methodology that has a solid social foundation and is an extension of the SE approach.

2.7 Multi-agent system development software

Agent platforms support effective design and construction of agents and multi-agent systems. An agent platform has to provide the following functions: agent management, agent mobility, agent communication, directory services (yellow pages), and interface to plug-in additional services. Figure 5 presents the architecture of a FIPA Agent Platform [3]. Several agent platforms are available: JADE (CSELT&Univ. of Parma, [45]), ZEUS (British Telecom, [44], [46]), AgentBuilder (Reticular Systems Inc., [47]), MadKit (LIRMM Montpellier [48], [49]), Volcano (LEIBNIZ, [40]), etc.

JADE (Java Agent Development framework) [45] is a free software framework for the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. JADE is written in Java language and is made by various Java packages, giving application programmers both ready-made pieces of functionality and abstract interfaces for custom, application dependent tasks. The main tools provided by JADE are the Remote Management Agent (RMA), the Dummy Agent, the Sniffer agent, the Introspector Agent, the SocketProxyAgent, the

DF GUI (a complete graphical user interface that is used by the default Directory Facilitator). The latest available version is JADE 3.1 (Dec. 2003).

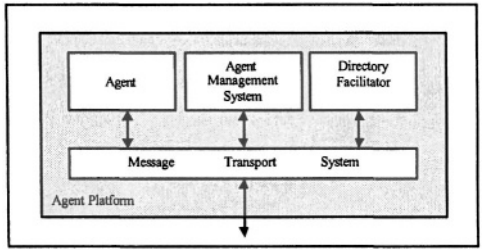

Figure 5. The architecture of a FIPA Agent Platform.

ZEUS is a generic, customisable, and scaleable industrial-strength collaborative agent toolkit. It consists in a package of classes implemented in Java, allowing it to run on a variety of hardware platforms [46]. The classes of the ZEUS toolkit are classified in three groups: an agent component library, an agent building tool, and an agent visualisation tool. ZEUS toolkit covers all stages of a MAS development, from analysis to deployment. It is limited to a single agent model.

AgentBuilder is an integrated software toolkit that allows software developers to quickly implement intelligent agents and agent-based applications [47]. The latest version is AgentBuilder 1.3 (based on Java 1.3), Windows XP compatible. Two versions are currently available: LITE, ideal for building single-agent, stand-alone applications and small agencies, and PRO, that has all the features of LITE plus an advanced suite of tools for testing and building multi-agent systems. AgentBuilder is grounded on Agent0/Placa BDI architecture. It is limited to a single agent model. Almost all stages of a MAS development are covered.

The MadKit toolkit provides a generic, highly customizable and scalable agent platform [48]. It is a generic multi-agent platform based on an organizational model called AGR (agent-group-role). MadKit is composed by a set of Java classes that implements the agent kernel, and various libraries of messages, probes and agents. Also, it includes a graphical development environment, system and demonstration agents. The MadKit micro-kernel is a small and optimized agent kernel which handles several tasks (control of local groups and roles, agent life-cycle management, and local message passing). The kernel is extensible through "kernel hooks". MadKit has a good versatility and light methodology (no BDI).

Volcano is a multi-agent platform that is under development [40], and whose aims are to fulfill the criteria of completeness (e.g. inclusion of MAS analysis and design phases), applicability (e.g. the versatility of the platform) and complexity (e.g. more friendly user interface, reuse of the platform). It

is based on the Agents Environment Interactions Organisations (AEIO) MAS decomposition. In this framework, *agents* are internal architectures of the processing entities, the *environment* is composed by the domain-dependent elements for structuring external interactions between entities, the *interactions* are elements for structuring internal interactions between entities, and *organisations* are elements for structuring sets of entities within the MAS. Volcano has a full analysis-to-deployment chain, including an open library of models and intelligent deployment tools.

3. APPLICATIONS

As reported in [50], the main application domains of multi-agent systems are ambient intelligence, grid computing, electronic business, the semantic web, bioinformatics and computational biology, monitoring and control, resource management, education, space, military and manufacturing applications, and so on. Many researchers has applied agent technology to industrial applications such as manufacturing enterprise integration, supply chain management, manufacturing planning, scheduling and control, holonic manufacturing systems.

In order to support interoperability and to allow heterogeneus agents and MASs to work together, some infrastructures are needed. Most of the multi-agent system applications adopted ad-hoc designs for the MAS infrastructure. However, in the recent years some MAS infrastructures were proposed.

3.1 Multi-agent systems infrastructures

Each multi-agent system architecture has its specific features: agent registration, agent capability advertisements, strategy for finding agents, agent communication language, agent dialogue mediation, agent content language, default agent query preference, etc. As multi-agent systems are open, in complex applications, homogeneity cannot be achieved with respect to the MAS architecture specific features. Thus, interoperation mechanisms must be designed and implemented.

3.1.1 RETSINA

RETSINA (Reusable Task Structure-based Intelligent Network Agents) [51] multi-agent infrastructure has been developed at the Carnegie Mellon University in Pittsburgh, USA. It is composed by four different reusable agent types that can be adapted to different applications, the interface agents,

task agents, information/resource agents, and middle agents. A collection of RETSINA agents forms an open society of reusable agents that self-organize and cooperate in response to task requirements. The RETSINA framework was implemented in Java. It is constructed on the principle that all agents should communicate directly with each other, if necessary. Agents find each other through a Matchmaker agent, who does not manage the transaction between the two agents, it just allow the direct communication between the two agents.

RETSINA is an open MAS infrastructure that supports communities of heterogeneus agents. It does not employ any centralized control on the MAS, rather implements distributed infrastructural services that facilitate the relations between the agents instead of managing them.

The RETSINA-OAA InterOperator acts as a connection between two MASs with two radically different agent architectures: the RETSINA capability-based MAS architecture and SRI' Open Agent Architecture (OAA). Agents in the OAA system "speak" Prolog-based OAA ICL, while agents in RETSINA system use KQML. The two languages has very different syntactic and semantic structures. OAA is organized around an agent called the Facilitator, which manages all the communications between agents in such a way that OAA agents cannot communicate directly.

3.1.2 SICS

SICS MarketSpace [52] is an agent-based market infrastructure implemented in Java. The goal of SICS is to enable automation of consumer goods markets distributed over the Internet. It consists in an information model for participant interests, and an interaction model that defines a basic vocabulary for advertising, searching, negotiating and settling deals. The interaction model is asynchronous message communication in a simple speech act based language, the *Market Interaction Format* (MIL).

3.2 Application areas

We have selected various MAS application areas (which are not disjunctive) and for each area a brief presentation of some MASs developments (the majority being simulations or prototypes) is made. The general application domains selected for this presentation are resource management (ADEPT business management, FACTS telecommunication service, TeleMACS, Challenger, MetaMorphII, MACIV); manufacturing planning, scheduling and control (TELE TRUCK); monitoring, diagnosis and control (ARCHON energy management); electronic commerce

(Fishmarket, SARDINE, eMediator, SMACE, COM_ELECTRON), and virtual enterprise (VIRT_CONSTRUCT).

3.3 ARCHON's electricity transportation management application

Energy management is the process of monitoring and controlling the cycle of generating, transporting and distributing electrical energy to industrial and domestic customers. A Spanish company, Iberdrola, that works in the energy domain decided to develop a set of decision support systems (DSS) in order to reduce the operators' cognitive load in critical situations, and to decrease the response time for making decisions. The DSS were interconnected and extended using the ARCHON technology. In [53] it is discussed the problem of development and deployment of MASs in real world settings, and it is analysed under the ARCHON project and applied to electricity transport management. ARCHON provides a descentralised software platform which offers the necessary control and level of integration to help the subcomponents to work together. Each agent consists of an ARCHON Layer (AL) and an application program (Intelligent System - IS).

Seven agents are running on five different machines. The agents are: BAI (Black-out Area Identifier), CSI-D and CSI-R (pre-existing Control System Interface), BRS (Breaks and Relays Supervisor), AAA (Alarms Analysis Agent), SRA (Service Restoration Agent), and UIA (User Interface Agent). The BAI agent identifies which elements of the network are initially out of service. CSI is the application's front end to the control system computers and consists of two agents: CSI-D detects the occurrence of disturbances and preprocesses the chronological and non chronological alarm messages which are used by the agents AAA, BAI and BRS; and CSI-R detects and corrects inconsistencies in the snapshot data file of the network, computes the power flowing through it and makes it available to SRA and UIA. The BRS agent detects the occurrence of a disturbance, determine the type of fault, generates an ordered list of fault hypotheses, validates hypotheses and identifies malfunctioning equipment. The AAA agent has similar goals with BRS. The SRA agent devises a service restoration plan to return the network to a steady state after a blackout has occurred. The UIA agent implements the interface between the users and the MAS.

Due to parallel activation of tasks, efficiency is achieved. Reliability is increased because even if one of the agents break down, the rest of agents can produce a result (not the best one) that could be used by the operator.

The application is operational since 1994. The MAS gives better results because it takes multiple types of knowledge and data into account and

integrates them in a consistent manner. Also, the system is robust because there are overlapping functionalities which means that partial results can be produced in the case of agent failure. The system is open, so new agents could be added in an incremental manner.

3.4 ADEPT business process management application

An agent-based system developed for managing a British Telecom (BT) business process is presented in [54]. The business process consists in providing customers with a quote for installing a network to deliver a particular type of telecommunications service. The process is dynamic and unpredictable, it has a high degree of natural concurrency, and there is a need to respect departmental and organisational boundaries. In this process, the following departments are involved: the customer service division (CSD), the design divison (DD), the surveyor department (SD), the legal department (LD), and the organisations that provide the out-sourced service of vetting customers (VCs). In the multi-agent system, each department is represented by an agent, and all the interactions between them take the form of negotiations (based on a service-oriented negotiation model). All negotiations are centered on a multi-attribute object, where attributes are, for instance, price, quality, duration of a service.

3.5 FACTS telecommunication service management

In the FACTS telecommunication service management [55], the problem scenario is based on the use of negotiation to coordinate the dynamic provisioning of resources for a Virtual Private Network (VPN) used for meeting scheduling by end users. A VPN refers to the use of a public network (as is the Internet) in a private manner. This service is provided to the users by service and network providers. The multi-agent system consists in a number of agents representing the users (Personal Communication Agents), and the service and network providers.

3.6 Challenger

In [56] it is described a MAS for distributed resource allocation, *Challenger*. The MAS consists of agents which individually manage local resources and which communicate with one another in order to share their resources (e.g. CPU time) in an attempt to efficiently use of them. *Challenger* is similar to other market-based control systems as the agents act as buyers and sellers in a marketplace, always trying to maximize their own utility. Experimental results of using the MAS to perform CPU load

balancing in a network of computers (small sized networks, e.g. of maximum 10 machines) are presented in [56]. Challenger was designed to be robust and adaptive. It is completely descentralized and consist of a distributed set of agents that run locally on every machine in the network. The main agent behaviour is based on a market/bidding metaphor with the following four steps: job origination, making bids, evaluation of bids, and returning results. Several simulations were run, including learning behaviours of the agents in order to improve the performance of the whole system in some critical situations such as large message delays and inaccurate bids made by the agents.

3.7 Tele-MACS

Tele-MACS [57] applied a multi-agent control approach to the management of an ATM network. In telecommunications, Tele-MACS considers link bandwidth allocation and dynamic routing. A multi-layered control architecture has been implemented in Java. Tele-MACS consists of multiple layers of MASs, where each layer is defined to conduct the control of the network infrastructure to a certain level of competence.

3.8 TELE TRUCK

Real-life transport scheduling can be solved by a multi-agent system. Each resource could be represented as an agent and market algorithms are applied to find and optimize solutions. The TELE TRUCK system, presented in [58], can be applied to online dispatching in a logistics management node of a supply web, and uses telecommunication technologies (satellite, GPS, mobile phones). The truck drivers, trucks, (semi)-trailers are autonomous entities with their own objectives, and only an appropriate group of these entities can perform together the transportation task. Thus the whole problem can be modeled as a MAS. Each entity is an intelligent agent, and has its own plan, goal, and communication facilities in order to provide the resources for the transportation plans according to their role in the society. In the TELE TRUCK system different types of negotiation techniques are used for the allocation of transportation tasks in a network of shipping companies. In the case of vertical cooperation, i.e. the allocation of orders within one shipping company, the simulated trading algorithm is used for dynamic optimization, and also an extended contract net protocol is used for fast and efficient initial solution (e.g. one order can be split to multiple trucks). The simulated trading algorithm is a randomized algorithm that realizes a market mechanism where contractors attempt to optimize a task allocation by successively selling and buying tasks in several trading rounds. In the case of

horizontal cooperation, i.e. the order allocation across shipping companies, a brokering mechanism is used for short-term cooperation. The matrix auction is another negotiation technique that is used. This type of auction is truth revealing and applicable for the simultaneous assignment of multiple items or tasks to bidders. For example, in the case of orders assignment to the vehicles, a bidding procedure is used. The dispatch officer in the shipping company interacts with a dispatch agent. The dispatch agent announces the newly incoming orders via an extended contract net protocol.

3.9 MetaMorphII

The MetaMorphII system [59] enables the development of a mediator-based multi-agent architecture to support enterprise integration and supply chain management. A federated-based approach is applied. A manufacturing system is seen as a set of subsystems that are connected through special interface agents called facilitators or mediators. Each enterprise has at least one mediator. In the supply chain network, partners, suppliers and customers are connected through their mediators. Other levels of mediators can exist inside an enterprise. The coordination mechanisms that are used include communication between agents through facilitators. Local agents use a restricted subset of an Agent Communication Language (ACL) to inform facilitators about their needs and services they offer. Facilitators use this information as well as their knowledge of the global MAS network to transform local agents' messages and route them to other facilitators. In this way, local agents give a part of their autonomy to facilitators and in turn, the facilitators satisfy their requirements.

3.10 Fishmarket

The Fishmarket project conducted at the Artificial Intelligence Research Institute (IIIA-CSIC), Barcelona, developed an agent-mediated electronic institution [60], [61]. FM100 is a Java-based version of the Fishmarket auction house, that allows to define auction-based trading scenarios where goods can be traded using the classical auction protocols (Dutch, English, Vickrey, and First-price Sealed Bid). It has a library of agent templates written in Java, Lisp, and C.

Fishmarket is one of the most popular simulations of an agent-mediated auction house, which offers a convinient mechanism for automated trading due to the simplicity of the conventions used for the interaction when multi-party negotiations are involved, and to the fact that on-line auctions may successfully reduce storage, delivery or clearing house costs in the fish market. FM was designed for the Spanish fish market. One main advantage

3.13 MACIV

MACIV is a multi-agent system for resource management on civil construction companies, developed in Java as an academic prototype used for demonstration of negotiation techniques [65]. In order to achieve an adequate solution and take into account the specific characteristics of the problem, it was decided to adopt a decentralized solution based on multi-agent systems techniques. The agents behaviours were improved through reinforcement learning. A multi-criteria negotiation protocol is used for a society of buyers and sellers, where buyer agents represent human operators requesting for tasks to be executed and seller agents represent resources competing for being used for those task execution.

3.14 SMACE

SMACE [66] is a MAS for e-commerce that supports and assists the creation of customised software agents to be used in agent-mediated e-commerce transactions. It has been used for testing automated negotiation protocols, including those based on the continuous double auction that support a multi-issue approach in the bargaining process. Learning is also included as a capability that enhance the MAS performance. The agents may have any negotiation strategy. SMACE was implemented in Java and JATLite was used to provide the communication infrastructure.

3.15 COM_ELECTRON

COM_ELECTRON is a multi-agent system developed at University of Ploiesti [67], which is dedicated to second hand electronic products selling. It has been implemented in JADE as a simulation. For the shopping agents architecture we have used the SmartAgent architecture [68]. The role of a SmartAgent is to assist users while doing electronic shopping in the Internet. The shopping agent may receive proposals from multiple seller agents. Each proposal defines a complete product offering including a product configuration, price, warranty, and the merchant's value-added services. The shopping agent evaluates and orders these proposals based on how they satisfy its owner's preferences (expressed as multi-attribute utilities). He negotiates over a set of issues that describe the characteristics of a good (such as: type of processor, memory capacity, price, hard disk capacity, in the case of a second hand laptop).

The main purpose of the agent-mediated electronic commerce system COM_ELECTRON is the maximization of the profit viewed as an increased number of transactions and deals agreed after some rounds of bilateral negotiation. In order to achieve this purpose, the negotiation model that was adopted by an agent (buyer/seller) is the service-oriented negotiation model described in [29] extended with an adaptability component capability implemented by a feed-forward artificial neural network [69], that allows him to model the other agent's negotiation strategy, thus doing an adaptive negotiation in order to make a better deal. The adaptive negotiation model was included in the architecture of the SmartAgent [70]. Figure 6 describes the price negotiation context.

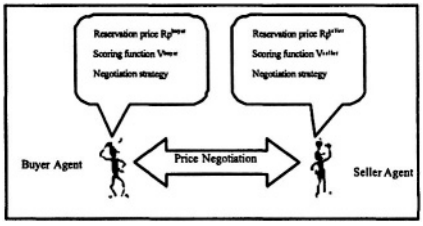

Figure 6. The price negotiation context.

This history of price proposals is a time series and the prediction of the next price proposal is made by using a feed forward artificial neural network. The learning capability is activated during the process of proposals and counterproposals exchanging and will influence the way in which the negotiation will evolve to an agreement. For example, the seller agent will reason about the buyer agent based solely on his observations of buyer's actions.

Currently, the COM_ELECTRON system uses a non-mediated interaction coordination model. If a coordination trace inspection it is needed, the architecture of the MAS can be modified by including some mediator agents which will capture the state of the interaction.

3.16 VIRT_CONSTRUCT

The agent-based virtual enterprise, VIRT_CONSTRUCT [71], [72], is under development at University of Ploiesti. The implementation of the system is made in JADE. The goal of VIRT_CONSTRUCT is the construction of private houses. Figure 7 presents the MAS view of VIRT_CONSTRUCT.

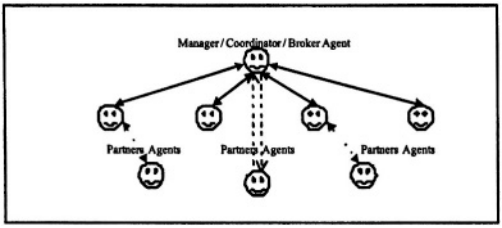

Figure 7. The MAS view of VIRT_CONSTRUCT.

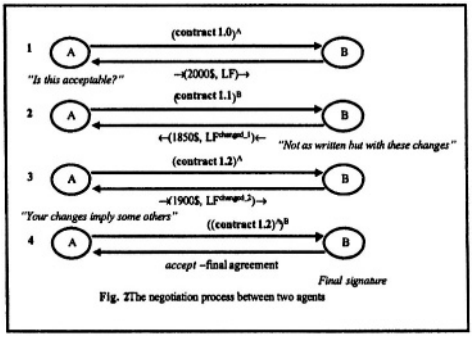

Figure 8. The negotiation process between two agents.

In the case of an agent-based VE, each partner is an agent that will act on behalf of the partner via delegation or in negotiation processes. In the context of an electronic marketplace, the creation of a VE involves the initiation of a competition between different agents that send bids in order to become the VE partners. Figure 8 describes an example of negotiation process between two agents, the Broker-Agent (A) and a potential partner's agent (B) that has the capability of specialized roof construction.

We have used two coordination mechanisms, the contract net protocol and a service-oriented negotiation model. Several simulations with different environment settings were run in JADE.

4. CONCLUSION

The multi-agent system approach has proved to be a proper solution for the design of complex, distributed computational systems. The main functionalities that a MAS has to provide are reasoning, communication, coordination, learning, planning, etc. Currently, the MAS developments have

ad-hoc designs, predefined communication protocols and scalability only in simulations. Therefore, problems with external non-agent legacy systems could arise. A lot of problems need to be solved. One of these is the lack of mature software development methodologies for agent-based systems. Currently, research work is focused in this direction, and the basic principles of software and knowledge engineering need to be applied to the development and deployment of multi-agent systems. Another is the need for standard languages and interaction protocols especially for their use in open agents societies. Also, there is a demanding need for developing reasoning capabilities for agents in open environments.

Summarizing, the problems that might appear when using a multi-agent system approach are the coordination in an open environment, the distributed resource allocation, the distribution of tasks, the agents interoperability, the privacy concerns, and the overall system stability.

From a software engineering point of view it is important to have some coordination tools that help engineers harnessing the intrinsic complexity of agent interaction by providing them with the most effective views on the state and evolution over time of interaction within the multi-agent system. Depending on the application domain that is modeled by a multi-agent system and on the specific type of negotiation needed at a specific moment, a distributive or an integrative negotiation model could be chosen. Usually, the integrative negotiation is applied to task/resource allocation, while for more complex domains, with multiple goals, an integrative negotiation model would be proper. An important aspect that need to be addressed when designing a multi-agent system is learning, which has the ability to improve the whole behaviour of a MAS.

Agent-based computing has the potential to improve the theory and the practice of modelling, designing, and implementing of complex systems. The main benefit of applying a multi-agent approach is that the partial subsystems can be integrated into a coherent and consistent super-system in which they work together to better meet the needs of the entire application.

References

1. N. R. Jennings, Agent-Based Computing: Promises and Perils, *Proceedings of the 16th International Joint Conference on Artificial Intelligence IJCAI99,* Stockholm, Sweden, pp. 1429-1436 (1999).
2. M. Wooldridge, Agent-based software engineering, *IEE Proceedings on Software Engineering,* **144**(1), 26-37 (1997).
3. FIPA (Foundation for Intelligent Physical Agents): http://www.fipa.org.
4. A. Newell, The Knowledge Level, *Artificial Intelligence,* **18**:87-127 (1982).
5. N. R. Jennings, and J. R. Campos, Towards a Social Level Characterisation of Socially Responsible Agents, *IEE Proceedings on Software Engineering,* **144**(1), 11-25 (1997).
6. L. C. Lee, H. S. Nwana, D. T. Ndumu, and P. de Wilde, The stability, scalability and performance of multi-agent systems, *British Telecom Journal,* **16**(3), 94-103 (1998).

7. M. Wooldridge, The theory of multi-agent systems, *lecture notes,* UPC-Barcelona (2000).
8. M. Wooldridge, and N. Jennings, Intelligent agent: theory and practice, *The Knowledge Engineering Review,* **10**(2), 115-152 (1995).
9. R. Brooks, Intelligence without representation, *Artificial Intelligence,* **47**(1-3), 139-159 (1991).
10. R. Sutton, Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming, *Proceedings of the 7th ICML,* Morgan Kaufmann (1990), 216-224.
11. A. Sloman, and B. Logan, Building Cognitively Rich Agents Using the SIM_AGENT Toolkit, *Communications of ACM* (1999).
12. KQML: http://www.cs.umbc.edu/kqml/.
13. M. Colombetti, A Commitment-Based Approach to Agent Speech Acts and Conversations, *Proceedings of Agent Languages and Conversation Policies workshop – Autonomous Agents 2000,* pp. 21-29 (2000).
14. H. Nwana, L. Lee, and N. Jennings, Coordination in Software Agent Systems, *BT Technology Journal,* **14**(4) (1996).
15. F. Bergenti, and A. Ricci, Three Approaches to the Coordination of Multiagent Systems, *Proceedings of the international ACM conference SAC2002,* Madrid, Spain (2002).
16. A. Omicini, and E. Denti, From tuple spaces to tuple centres, *Science of Computer Programming,* **42**(3):277-294 (2001).
17. R. Bourne, C. B. Excelente-Toledo, and N. R. Jennings, Run-Time Selection of Coordination Mechanisms in Multi-Agent Systems, *Proceedings of ECAI2000,* Berlin, Germany (2000).
18. E. de Jong, Multi-Agent Coordination by Communication of Evaluations, *technical report,* Artificial Intelligence Laboratory, Vrije Universiteit Brussel (1997).
19. M. Barbuceanu, and M. S. Fox, Integrating Communicative Action, Conversations and Decision Theory to Coordinate Agents, *ACM Proceedings of Autonomous Agents 97,* Marina Del Rey, California, USA, pp. 49-58 (1997).
20. S. Ossowski, A. Garcia-Serrano, and J. Cuena, Emergent Co-ordination of Flow Control Actions through Functional Co-operation of Social Agents, *Proceedings of the 12th European Conference on Artificial Intelligence-ECAI96,* Budapest, Hungary, pp. 539-543 (1996).
21. A. Bazzan, Evolution of Coordination as a Metaphor for Learning in Multi-Agent Systems, *Proceedings of the ECAI96 workshop W26,* Budapest, Hungary (1996).
22. P. P. Grassé, La theorie de la stigmergie: essai d'interpretation du comportament des termites construeteurs, Insectes Sociaux 6 (1959).
23. G. Theraulaz, G., A brief history of Stigmergy, *Artificial Life,* **5**, pp. 97-116 (1999).
24. P. Valckenaers, H. Van Brussel, M. Kollingbaum, and O. Bochmann, Multi-agent Coordination and Control Using Stigmergy Applied to Manufacturing Control, *Multi-Agent Systems and Applications,* Lecture Notes in Artificial Intelligence – LNAI 2086, Springer (2001).
25. D. G. Pruitt, *Negotiation Behavior,* Academic Press, New York (1981).
26. M. Fisher, Characterising Simple Negotiation as Distributed Agent-Based Theorem-Proving – A Preliminary Report, *Proceedings of the International Conference on Multi-Agent Systems* – ICMAS, pp. 127-134 (2000).
27. S. Kraus, Automated Negotiation and Decision Making in Multiagent Environments, M. Luck et al. (Editors), *Multi-Agent Systems and Applications,* LNAI 2086, (2001), 150-172.
28. M. Oprea, Adaptability in Agent-Based E-Commerce Negotiation, *tutorial notes* of the 20th IASTED International Conference Applied Informatics AI'02 – symposium Artificial Intelligence Applications–AIA'02, February, Innsbruck, Austria (2002).

29. P. Faratin, C. Sierra, and N. R. Jennings, Negotiation decision functions for autonomous agents, *Robotics and Autonomous Systems,* **24**:159-182 (1998).
30. K. P. Sycara, Persuasive argumentation in negotiation, *Theory and Decision,* **28**:203-242 (1990).
31. S. Kraus, K. Sycara, and A. Evenchik, Reaching agreements through argumentation: a logical model and implementation, *Artificial Intelligence,* **104**(1-2), 1-69 (1998).
32. S. Parsons, C. Sierra, and N. R. Jennings, Agents that reason and negotiate by arguing, *Journal of Logic and Computation,* **8**(3), 261-292 (1998).
33. S. Kraus, *Strategic Negotiation in Multiagent Environments,* MIT Press, Cambridge, USA (2001).
34. M. Oprea, Adaptability and Embodiment in Agent-Based E-Commerce Negotiation, *Proceedings of the Workshop Adaptability and Embodiment Using Multi-Agent Systems*-AEMAS01, Prague, (2001) 257-265.
35. S. Sen, and M. Sekaran, Individual Learning of coordination knowledge, *Journal of Experimental & Theoretical Artificial Intelligence,* **10**(3), 333-356 (1998).
36. A. Pérez-Uribe, and B. Hirsbrunner, The Risk of Exploration in Multi-Agent Learning Systems: A Case Study, *Proceedings of the Agents-00/ECML-00 workshop on Learning Agents,* Barcelona, (2000), 33-37.
37. D. Zeng, D., and K. Sycara, How Can an Agent Learn to Negotiate, *Intelligent Agents III. Agent Theories, Architectures and Languages,* LNAI 1193, Springer, 233-244, (1997).
38. D. Carmel, and S. Markovitch, Model-based learning of interaction strategies in multi-agent systems, *JETAI* **10**, 309-332 (1998).
39. J. Vidal, and E. Durfee, Learning nested agent models in an information economy, *Journal of Experimental & Theoretical Artificial Intelligence,* **10**(3), 291-308 (1998).
40. Y. Demazeau, Multi-Agent Methodology and Programming, *tutorial notes,* ACAI'2001 & EASSS'2001, Prague (2001).
41. F. Bergenti, O. Shehoty, and F. Zambonelli, Agent-Oriented Software Engineering, *tutorial notes,* EASSS2002, Bologna (2002).
42. AUML: http://auml.org
43. B. Bauer, J. P. Müller, and J. Odell, Agent UML: A Formalism for Specifying Multiagent Interaction, *Agent-Oriented Software Engineering,* P. Ciancarini and M. Wooldridge (Eds.), Springer, Berlin, pp. 91-103 (2001).
44. H. S. Nwana, D. T. Ndumu, L. Lee, and J. C. Collis, A Toolkit for Building Distributed Multi-Agent Systems, *Applied Artificial Intelligence Journal,* **13**(1) (1999) - Available on line from http://www.labs.bt.com/projects/agents.
45. JADE (Java Agent Development Framework), http://jade.cselt.it/.
46. J. C. Collis, and L. C. Lee, Building Electronic Commerce Marketplaces with ZEUS *Agent Tool-Kit, Agent Mediated Electronic Commerce,* P. Noriega, C. Sierra (Eds.), LNAI 1571, Springer, (1999), pp. 1-24.
47. AgentBuilder: http://www.agentbuilder.com.
48. MadKit: http://www.madkit.org.
49. O. Gutknecht, and J. Ferber, MadKit: A generic multi-agent platform, *Proceedings of the Fourth International Conference on Autonomous Agents – AA2000,* Barcelona, (2000), pp. 78-79.
50. M. Luck, P. McBurney, and C. Preist, Agent Technology: Enabling Next Generation Computing – A Roadmap for Agent Based Computing, AgentLink II (Jan. 2003).
51. K. Sycara, Multi-agent Infrastructure, Agent Discovery, Middle Agents for Web Services and Interoperation, *Multi-Agent Systems and Applications, M.* Luch et al. (Eds), LNAI 2086, Springer, (2001),pp. 17-49.

52. J. Eriksson, N. Finne, and S. Janson, SICS MarketSpace – An Agent-Based Market Infrastructure, *Agent Mediated Electronic Commerce,* LNAI 1571, Springer, pp. 41-53 (1999).
53. N. R. Jennings, J. M. Corera, I. Laresgoiti, Developing Industrial Multi-Agent Systems, *Proceedings of ICMAS* (1995), pp. 423-430.
54. N. R. Jennings, P. Faratin, M. J. Johnson, T. J. Norman, P. O'Brien, and M. E. Wiegand, Agent-based business process management, *International Journal of Cooperative Information Systems,* **5**(2&3):105-130 (1996).
55. FACTS (1998) http://www.labs.bt.com/profsoc/facts.
56. A. Chavez, A. Moukas, and P. Maes, *Challenger:* A Multi-agent System for Distributed Resource Allocation, Proceedings of Autonomous Agents 97, Marina Del Rey, USA, (1997), pp. 323-331.
57. Tele-MACS: http://www.agentcom.org/agentcom/.
58. H. –J. Bürckert, K. Fisher, and G. Vierke, Transportation scheduling with holonic MAS – the teletruck approach, *Proceedings of the 3d International Conference on Practical Applications of Intelligent Agents and Multiagents - PAAM'98,* UK (1998).
59. W. Shen, Agent-based cooperative manufacturing scheduling: an overview, *COVE Newsletter,* No. 2, (March 2001).
60. Fishmarket, http://www.iiia.csic.es/Projects/fishmarket/newindex.html.
61. P. Noriega, Agent-Mediated Auctions: The Fishmarket Metaphor, *PhD Thesis,* Artificial Intelligence Research Institute-IIIA-CSIC, Barcelona (1997).
62. J. Morris, P. Ree, and P. Maes, Sardine: Dynamic Seller Strategies in an Auction Marketplace, *Proceedings of the International Conference Electronic Commerce,* Minneapolis, Minnesota, USA, ACM Press, (2000).
63. eMediator, http://www.ecommerce.cs.wustl.edu/eMediator.
64. T. Sandholm, *eMediator.* A Next Generation Electronic Commerce Server, *Proceedings of the 4th International Conference Autonomous Agents,* Barcelona, ACM Press, (2000), pp. 341-348.
65. J. M. Fonseca, A. D. Mora, and E. Oliveira, MACIV: A Multi-Agent System for Resource Selection on Civil Construction Companies, *Technical Summaries of the Software Demonstration Session – in conjunction with Autonomous Agents'00* (2000).
66. H. L. Cardoso, and E. Oliveira, SMACE, *Technical Summaries of the Software Demonstration Session – in conjunction with Autonomous Agents'00* (2000).
67. M. Oprea, COM_ELECTRON a multi-agent system for second hand products selling – a preliminary report, *research report,* University of Ploiesti (2003).
68. M. Oprea, The Architecture of a Shopping Agent, *Economy Informatics,* **II**(1), 63-68 (2002).
69. M. Oprea, The Use of Adaptive Negotiation by a Shopping Agent in Agent-Mediated Electronic Commerce, *Multi-Agent Systems and Applications III,* LNAI 2691, Springer, 594-605 (2003).
70. M. Oprea, An Adaptive Negotiation Model for Agent-Based Electronic Commerce, *Studies in Informatics and Control,* **11** (3), 271 -279 (2002).
71. M. Oprea, Coordination in an Agent-Based Virtual Enterprise, *Studies in Informatics and Control,* **12**(3), 215-225 (2003).
72. M. Oprea, The Agent-Based Virtual Enterprise, *Journal of Economy Informatics,* **3**(1), 21-25 (2003).