Methodological Issues for Designing Multi-Agent Systems with Machine Learning Techniques: Capitalizing Experiences from the RoboCup Challenge

A. Drogoul & J.-D. Zucker

LIP6 - Université Paris 6

Boîte 169 - 4 Place Jussieu

75252 PARIS CEDEX 05

{Alexis.Drogoul, Jean-Daniel.Zucker}@lip6.fr

Abstract

This paper deals with one of the probably most challenging and, in our opinion, little addressed question that can be found in Distributed Artificial Intelligence today, that of the methodological design of a learning multi-agent system (MAS). In previous work, in order to solve the current software engineering problem of having the ingredients (MAS techniques) but not the recipes (the methodology) we have defined *Cassiopeia,* an agent-oriented, role-based method for the design of MAS. It relies on three important notions: (1) independence from the implementation techniques; (2) definition of an agent as a set of three different levels of roles; (3) specification of a methodological process that reconciles both the bottom-up and the top-down approaches to the problem of organization. In this paper we show how this method enables Machine Learning (ML) techniques to be clearly classified and integrated at first hand in the design process of an MAS, by carefully considering the different levels of behaviors to which they can be applied and the techniques which appear to be best suited in these cases. This presentation allows us to take a broad perspective on the use of all the various techniques developed in ML and their potential use within an MAS design methodology. These techniques are illustrated by examples taken from the RoboCup challenge. We then show that a large part of the design activity is nevertheless left to be done as a result of heuristic choices or experimental work. This allows us to propose the *Andromeda* framework, which consists in a tighter integration of ML techniques *within Cassiopeia* itself, in order to assist the designer along the different steps of the method and to develop self-improving MAS.

Keywords: agent-oriented design, distributed machine learning, RoboCup Challenge, collective robotics.

1. INTRODUCTION

Multi-Agent Systems (MAS) have never been technically so simple to implement and conceptually so difficult to design. A number of frameworks are available, theoretical studies are increasing continuously, and it is now even possible to find commercial products that propose to solve almost any of the problems a software engineer usually faces when building an MAS: agents' architectures, cooperation and coordination algorithms, communication languages, etc.

The attractiveness of MAS has also increased since its birth: these systems, which used to be restricted to specific areas of application in the late eighties (Bond and Gasser, 1988), can now be found almost everywhere, from natural language processing to planning or problem solving. The potential to apply a promising "divide and conquer" approach to problems that were once thought to be solvable only by centralized techniques (Decker, 1987; Drogoul and Dubreuil, 1991); the ability to use sociological or even biological rather than the traditional psychological metaphors when designing an intelligent system (Drogoul and Ferber, 1993; Korf, 1992); the continuous success of internet-related systems and applications which "naturally" require a decentralized scheme (Moukas, 1996); the appeal of building open systems (Hewitt and Inman, 1991) rather than closed ones, everything, from marketing to research, contributes to make Distributed AI one of the most promising trends in the present - and future - of AI.

However, ten years after the first use of the term "DAI" for describing this broad domain of research that extends from distributed systems to Artificial Life (Demazeau and Muller, 1991; Werner and Demazeau, 1992), we claim (and intend to demonstrate in this paper) that little has been done on the most delicate part of this novel approach. Although MAS are now used in very large software engineering projects (Wooldridge, 1997), there is no methodology available so far, for allowing software engineers to use the MAS technology in routine operations. And this claim is even more true when one tries to build adaptive MAS - i.e. MAS that can learn how to improve themselves using Machine Learning techniques.

1.1. What is a methodology?

Providing a platform, even splendidly programmed, for designing a system is like providing ingredients without a recipe: depending on the skills of the cook, there is always a chance that something good will be cooked, but maybe not. However, the cook certainly prevents himself from reproducing this experiment, changing it a little bit to accommodate different tastes or teaching it to somebody else. It is and remains a one-shot, stand-alone, success! Methodologies in computer science are what recipes are for cooking: they replace neither the intuition nor the coding, but provide a framework for unifying these two activities.

Basically, the main role of a methodology is to identify the steps that are necessary to proceed from the definition of the requirements of a project to their fulfillment (i.e. the project life cycle). More concretely, a methodology supplies the tools for transforming an always intuitive and subjective vision of a system to be built (the client's requirements, for instance) into a formalized and objective (i.e. which can be shared and reused) definition of the same system once it has been implemented. It thus provides a project with "something" that will stand and remain somewhere between the original blueprint and the final code:

1. A structured set of guidelines, which includes the steps mentioned above, advice for each of these steps, and how to proceed from one step to another.

2. A unified way to document the design process. This is used for sharing the experience gained during this process among designers and/or across time whenever, for instance, the design has to be undertaken by other software engineers.

3. The use of a homogeneous terminology, which has a meaning at each step of the cycle and supports the transitions from step to step (it usually also includes a graphical terminology based on diagrams and flowcharts).

4. The use of *operational* conceptual abstractions; that is, conceptual structures abstract enough to allow a sufficient choice of techniques when it comes to implementing the system, but operational enough to prevent the designer from using unrelated or outdated techniques.

5. A comprehensive and incremental history of the project, which gives the possibility to backtrack from any step to previous ones without losing what has been done before.

1.2. Why a methodology?

In the case of Distributed Artificial Intelligence, the project requirements intuitively consist in having a number of agents achieving a collective task. To fulfill these requirements, one must design the agents with specific attention to both their skills (or functional faculties) and their abilities to organize themselves. This implies managing at least three different levels of abstraction at the same time:

1. The level of the individual agents (What architecture to implement them? What is their knowledge and how do they manage it? What are their skills and how are they distributed among the agents?).

2. The level of the interactions between the agents (How and what do they communicate? Can they act on each other and in which way?).

3. The level of their organization (how do they cooperate? Is there a shared goal and how can they plan to collectively reach it? What is the structure of the organization and how does it evolve?)

The RoboCup challenge (Kitano et al., 1997a; Kitano et al., 1995) is a good example of this necessity: designing a team of soccer-playing robots requires the designer to pay attention simultaneously to these three levels. The programming of the individual skills is as important as the design of the inter-individual coordination mechanisms or that of the collective cooperation schemes. Moreover, there is an important interplay between these three levels: for example, knowing how to dribble can enable the players to form powerful collective combinations; on the contrary, being provided with a team strategy may require the players to undertake new individual responsibilities (i.e. special tactics not specifically required when playing alone).

The existing methodologies, especially the object-oriented methodologies (Graham, 1994) that can be considered because of some similarities such as distribution or locality[1], provide an interesting basis of analysis (Abbott, 1983; Coad and Yourdon, 1991) since they enable the distribution of the initial requirements along the structural and behavioral axes. However, they do not offer any methodological framework for taking the various organizational issues into

[1] And also because most agent-based systems are programmed using object-oriented languages.

account, because the organization is not considered as an object of analysis by itself (Booch, 1994). A number of very interesting studies on building agent-oriented methodologies as extensions of object-oriented ones have nevertheless been undertaken (see for example (Pont and Moreale, 1996)), but what they can really provide so far is still unclear: agents and objects differ in a number of important respects, the most noticeable being the ability of agents to dynamically and autonomously change their organization, something that objects are not supposed to be allowed to do. We believe that although object-oriented languages are the prime target for implementing MAS, it would be an error to consider agents as only "super-objects" (see for instance Wooldridge's arguments about the importance of the *intentional stance* for building MAS (Wooldridge, 1997) even if, in his case, he still relies on a specific technique, the BDI architecture, for designing an MAS).

On the other hand, the little DAI work (e.g. (Moulin and Cloutier, 1994; Wooldridge, 1992)) that deals with methodological aspects either indirectly addresses the organizational issues through the use of specific negotiation or coordination techniques, or imposes certain agents' architectures - which in fact are only particular methods of implementation. The *Cassiopeia* method, presented in (Collinot and Drogoul, 1998; Drogoul and Collinot, 1998), is an attempt to overcome this problem of reliance on the implementation by providing a role-based abstract view of the agents within which the different techniques can easily be classified. Like object-oriented methods, it does not solve the problem of design but provides a framework for explicitly and methodically expressing the hypotheses and choices that are being made during the design process.

1.3. Learning Multi-Agent Systems (LMAS)

The design requirements expressed above are even stronger when the goal is to make adaptive MAS, by introducing Machine Learning (ML) techniques that allow the agents to learn how to behave, how to interact or how to organize themselves (Weiss and Sen, 1995). Indeed, it requires the agents to learn different abilities at different levels of abstraction. And it is important not to confuse these levels: individually learning a given skill is not at the same level as collectively learning how to counteract a strategy, for example. Moreover, from a technical point of view, the hypotheses, the protocols and the ML techniques to be used in these two cases are not likely to be the same.

In this respect, the introduction of ML techniques in MAS needs to be undertaken methodologically, carefully identifying which techniques to use at which level and which levels to consider. Although this introduction has received a great deal of attention in recent years (Sen, 1997; Weiss, 1996), few people have really considered the specificity of MAS compared to traditional ML systems. The majority of the studies deal with very narrow subjects such as the suitability of given ML techniques to such and such task, but none of them really handles **all** the following questions, which are central to the design of LMAS:

1. *Beneficiary*: When an agent learns something, who is the beneficiary? The agent itself? The group of agents? The overall organization? All of them?
2. *Learning process*: How does it learn? By itself? In interaction with other agents? In a group?
3. *Learning tasks*: What does it learn? How to behave? How to interact? How to organize itself with the others?
4. *Learning techniques*: Which technique to use for a given task and a given process? How to compare their suitability?
5. *Learning protocols*: What is the context that would allow for a correct evaluation of the learning task?

We claim that these questions, among others, cannot be answered without being asked within an MAS design methodological process, which allows the designer to deal with these multiple levels of abstraction while providing a framework for possibly revising the choices. Conversely, we strongly believe that the introduction of ML techniques within the methodological tool itself is the only way to solve the usual problems one faces when designing an MAS (be it adaptive or not).

The most crucial of these problems are pointed out at the end of Section 2, after the presentation of the *Cassiopeia* methodology and with the help of a short presentation of the team we have designed for the RoboCup simulation league. We then propose a classification of different machine learning techniques and concepts in Section 3. The *Andromeda*[2] methodology, which is a result of a tight integration of learning methods with the *Cassiopeia* concepts, is introduced in Section 4. We conclude by providing some hints about the future of our attempt to build a truly agent-based computer-aided MAS design system.

2. CASSIOPEIA: AN AGENT-ORIENTED, ROLE-BASED MAS DESIGN METHODOLOGY

2.1. Overview of *Cassiopeia*

The *Cassiopeia* method is a way to address a type of problem-solving where collective behaviors are put into operation through a set of agents. It is not targeted at a specific type of application nor does it require a given architecture of agents. However, it is assumed that although the agents can have different aims the goal of the designer is to make them behave cooperatively. *Cassiopeia* relies on several concepts, namely those of **role**, **agent**, **dependency**, and **group.** The main idea is that we view an agent as nothing else but a set of roles, among which we distinguish three levels (see figure 1):

1. **Individual roles**, which are the different behaviors that the agents are individually able to perform, regardless of the policy they will choose to perform them with.
2. **Relational roles**, that is how they choose to interact with one another (by enabling/disabling individual roles), with respect to the mutual dependencies of their individual roles.
3. **Organizational roles**, or how the agents can manage their interactions to become or stay organized (by enabling/disabling some relational roles).

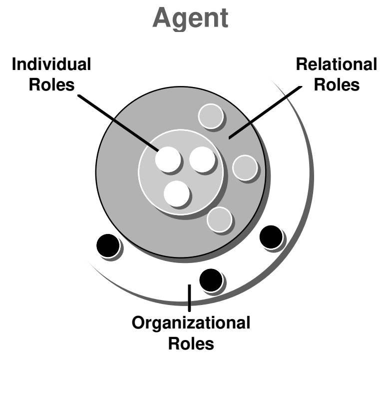

Figure 1 - An abstract view of an agent with its three levels of roles

[2] We have called the method Andromeda, the daughter of Cassiopeia in Greek mythology.

Cassiopeia proceeds from the definition of the collective task to the design of the MAS along five steps, depicted in figure 2 as layers, that reconcile both the local and global views of an MAS (see figure 3):

1. **The individual roles layer,** which contains the definition of the required individual roles in order to define the types of agent.

2. **The dependencies layer,** which contains the definition of the dependencies between these roles (functional, resource-based or goal-based dependencies).

3. **The relational roles layer,** which contains the definition of the way the agents can handle these dependencies, by playing given relational roles that enable them to influence other agents or choose how to be influenced by them.

4. **The groups layer,** which contains the definition of the potential groups that may appear, with respect to the above influences.

5. **The organizational roles layer,** which contains the description of the dynamics of these groups, that is the organizational roles the agents have to play to make them appear, evolve or disappear.

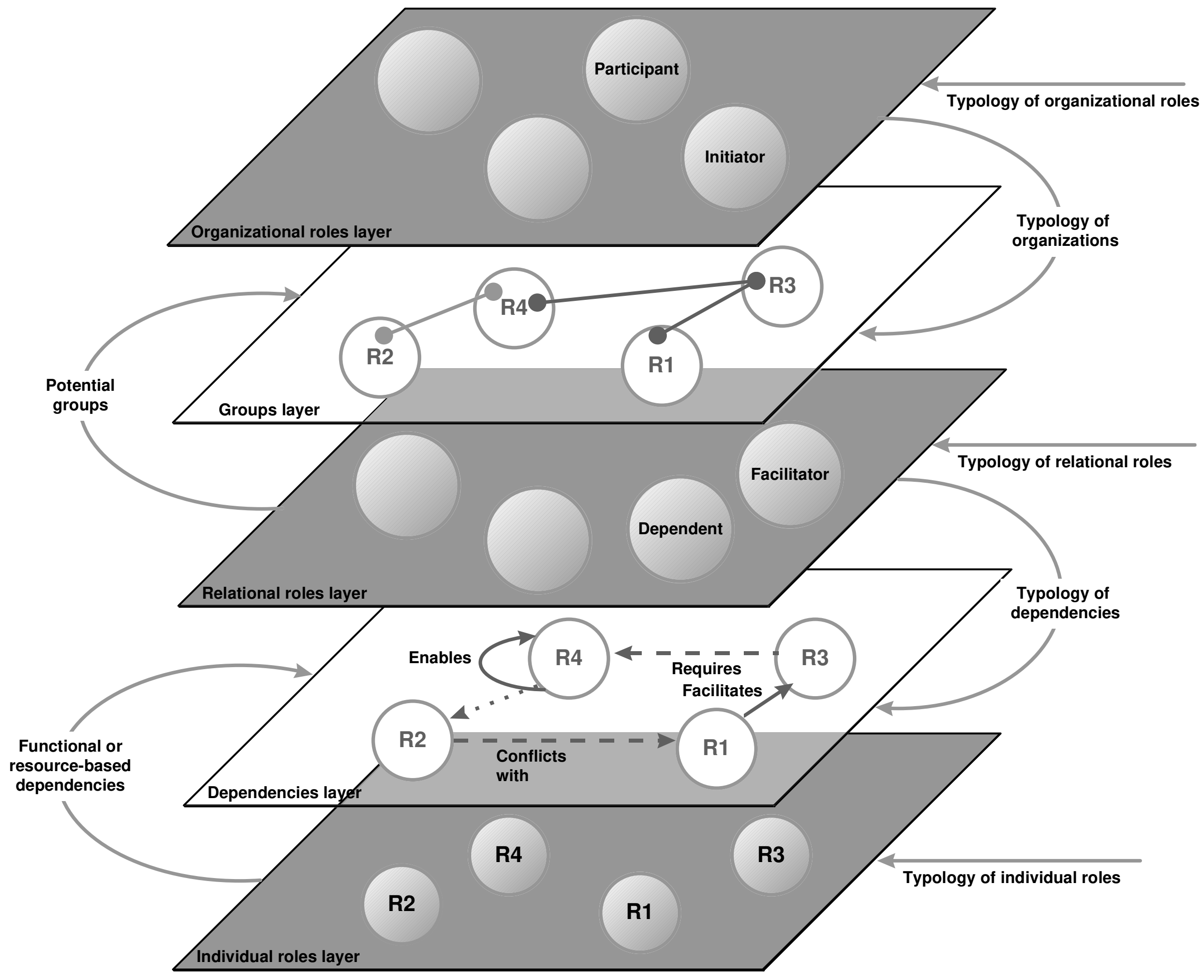

Figure 2 - The five layers used in Cassiopeia. The relational and organizational layers depicted here have deliberately been simplified (with two roles in each).

The order in which the five layers must be designed is not prescribed, in order for methodology to accommodate either a top-down or a bottom-up approach (or a mixture of the two). However, the usual way to enter the method is to begin by the individual roles layer and to end by the organizational roles layer, as depicted in figure 3. It is a bottom-up approach, but not in the commonly held sense of the term "bottom-up", because the overall organization (in the definition of the roles) is taken into account from the beginning. Moreover, and unlike most approaches to the design of groups of agents (for example, Mataric's one (Mataric, 1993)), the process is not intended to be sequential, but iterative and incremental.

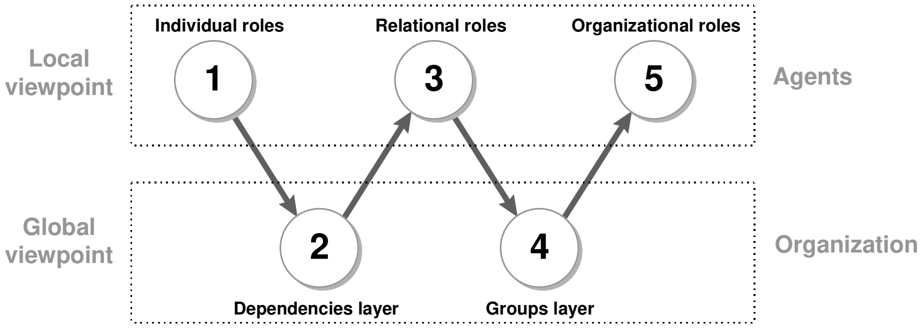

Figure 3 - Cassiopeia's design steps and the top-down and bottom-up paths between them.

2.2. Step 1 - definition of the individual roles

The identification of the individual behaviors that should be put into operation by the agents to achieve the task being considered does not come under *Cassiopeia*: most of the time, it results from a functional (e.g. (Yourdon, 1989)) or object-oriented (e.g. (Abbott, 1983; Coad and Yourdon, 1991; Rumbaugh et al., 1991)) analysis step. Given these behaviors, the first step of Cassiopeia consists in identifying the required level of abstraction so that the behaviors make sense with respect to the collective achievement of the task: the designer specifies the sets of behaviors that achieve proper functionalities, and thus determines the individual roles that the agents can play.

In most of the cases, the definition of the appropriate roles proceeds in an iterative fashion, by combining a bottom-up approach — the roles as sets of behaviors — with a top-down approach — the roles as an integral part of the organization to be achieved. The notion of role is an ambivalent concept, which represents both the function an agent is achieving at a given time and the position it occupies at the same time in the group of agents. This role definition requires special care, in that the subsequent design operations will largely rely on it.

The designer is free to define different types of agents based on these roles and can choose to design agents that are either homogeneous (all the agents are provided with the same set of domain-dependent roles) or heterogeneous (some agents are supplied with only a subset of these roles).

2.3. Steps 2 & 3 - dependencies and relational roles

The second and third steps consist in analyzing the structure of the organization based on the dependencies between the individual roles being considered. Such dependencies can be functional, when they derive from the dependencies that exist between the behaviors implemented by these roles, or relational, when they take place at the abstraction level of the roles. Various types of dependencies can be considered such as coordination, simultaneous or sequential facilitation or conditioning (see (Collinot and Drogoul, 1998)). The typology to be used is not *a priori* defined by *Cassiopeia* but should be introduced at the level of the relational roles. The identification of these dependencies leads to defining the *dependencies layer* of the collective task being considered. The designer removes the inconsistent dependencies and when necessary can decide to ignore some dependencies according to the available heuristics of the application domain. At this point, the graph only contains the dependencies between roles that are supposed to be relevant with respect to the collective achievement of the task.

The dependencies between the individual roles are naturally translated into dependencies between the agents that can play these roles. In order for the agents to determine their role depending on those played by the other agents, they must be provided with some means for identifying and handling these dependencies. In this aim, *Cassiopeia* resorts to the abstract notion of *influence*: an influence relationship between an agent A and an agent B relies on an existing dependency between the role played by A and the role played by B. A dependency relationship gives rise to two distinct roles, the role of *influencing agent* and the role of *influenced agent*, which are put together under the term of *relational roles* (the names of the roles depend of course on the dependency being considered: for example, the *inhibition* dependency would give rise to the roles of *inhibitor* and *inhibited*). An influencing agent produces *signs of influence* (anything that enables it to communicate it or be perceived as playing this role) that correspond to the individual role it is playing; an influenced agent must be able to interpret these signs to play the individual role that corresponds to the influence that has been exerted. The definition of these relational roles therefore makes it possible to distribute the structure of the organization among the agents, as a transmission of influence signs. At this step, influence signs that come from other sources than the agents, as for example those produced by the environment, can also be taken into account.

When setting the relational roles, the designer must specify the behaviors to control the individual roles (for instance, how to choose among several influences).

2.4. Step 4 & 5 - groups and organizational roles

At this time, all the potential groups of agents have been defined implicitly. The fourth step consists in defining the potential groups that may arise. The paths and the elementary circuits of the dependencies layer can be used to define the potential groupings of the different individual roles therefore provide a global representation of the structure of the organizations to which the agents can belong when playing these roles. This is what the groups layer depicts.

The fifth and last step addresses the dynamics of the organization. It consists in specifying the *organizational roles* that will enable the agents to manage the formation and dissolution of the defined groups. A basic typology that lists two roles - *initiator*, *participant* - is defined in *Cassiopeia*, but it is possible to define new roles (which will have an impact on the definition of the groups by providing new ways of forming groups - i.e. non-hierarchical groups, etc.).

As described in the previous step, when an agent is involved in an influence relationship and produces some influence signs, it adopts a *role of initiator* since it initiates the formation of a group with other agents (see figure 4). An initiator agent plays an individual role that makes it belong to all the groups it can potentially form with the agents playing roles it is currently influencing. As a consequence, it can evaluate them to decide which agents are the most appropriate to form a group in the current context. The designer thus (1) identifies the agents likely to play an initiator role (according to the grouping potential that has been identified in the dependencies layer), and (2) determines, for each of them, the selection methods that allow it to control the formation and dissolution of groups. For an agent, playing the role of initiator consists in activating *group formation behaviors* as well as *group dissolution behaviors*. Similarly, playing the role of a participant consists in activating *commitment behaviors*.

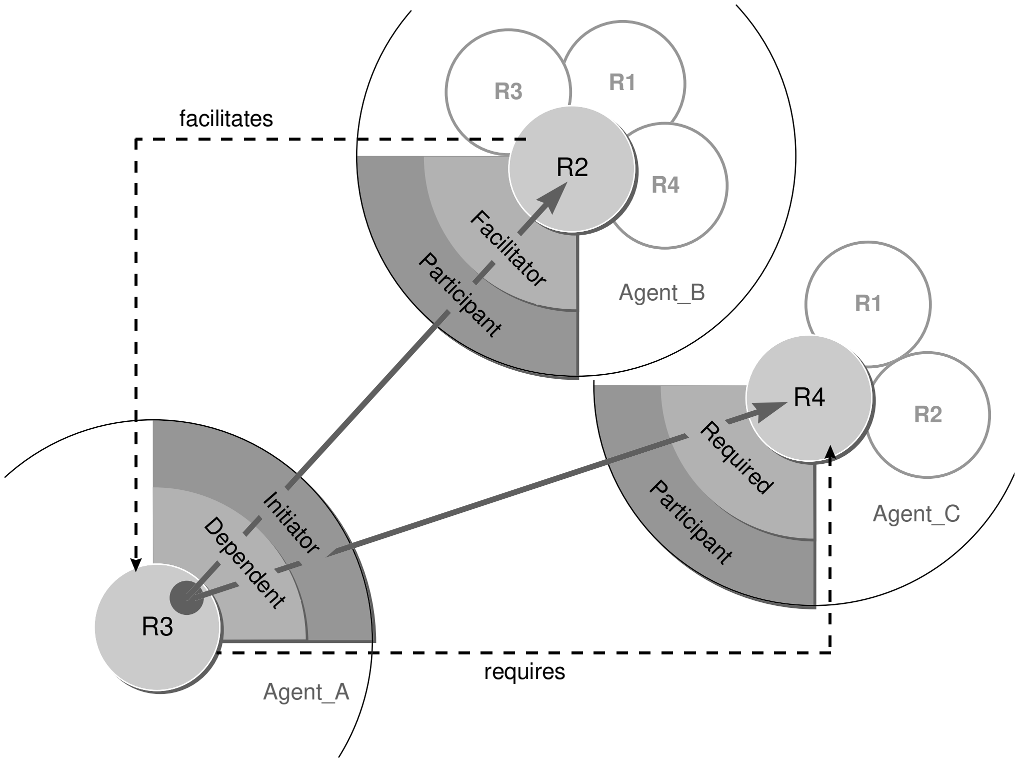

Figure 4 - Example of a group formation.
White circles represent the inactive roles

Group formation behaviors are specified to enable an initiator agent to dynamically organize collaboration with other agents. Basically, these behaviors are aimed at selecting the most appropriate agents to collaborate with. These agents are then going to play a role of participant in this group. Next, the designer specifies the *commitment sign*s, which enable the initiator agent to indicate to the selected agents that a group is formed with them. The designer must thus specify the commitment behaviors used by the agents to control their relational roles in response to the commitment signs. It is possible, for instance, to consider that these agents should inhibit all or part of their relational roles so as to remain committed to the group — as long as it is maintained. Finally, the designer must consider the dissolution of a group. A group ceases to exist when the agent that plays the role of initiator is satisfied; or a group can be replaced by a group that is considered as more efficient. The designer must thus define the group dissolution behaviors which produce dissolution signs that are interpreted by the commitment behaviors of the participant agents.

The table in figure 5 sums up the three levels of roles that are defined by *Cassiopeia*, along with the different types of signs and associated behaviors.

	Roles	Basic typology*	Associated behaviors	Exchanged signs
Agent	**individual**	*application-dependent*	*application- dependent*	–
	relational	influencing agent	producing the influence signs according to the domain role	influence signs
		influenced agent	interpreting the influence signs in order to control the individual roles	
	organizational	initiator	group formation behavior	dissolution signs
			group dissolution behavior	
		participant	commitment behavior	commitment signs

*The exact typology can be more complex, if one wants to consider specific influences or different ways to create groups.

Figure 5 - Cassiopeia in a nutshell

2.5. Main methodological problems

Cassiopeia has been used to design a number of teams of soccer-playing (simulated) robots that run on the SoccerServer (Kitano et al., 1997a). Obviously, in such a typical team-game the aim of the designer is to enable her or his team to score goals while defending its own side. Thus, the problem is to design the agents so that they can play with what we might call a "team spirit", that is a set of explicit or implicit social rules and constraints that allows them to play collectively and prevents them from disturbing their team mates while they are playing. Consequently, in addition to their individual soccer-playing skills, the agents should have abilities to organize themselves in order to win. From the standpoint of multi-agent systems, the main difficulty is then to express *locally* (i.e. at agent level) the behaviors which allow the *collective* achievement of the task by the team to be obtained. Two other difficulties make this game a very interesting, though very challenging, standard problem for multi-agent systems:

1. The dynamics of the game makes it impossible both to define the organization of the robots in advance and to control the game in a centralized way.

2. The operations of the opposing team are by definition unpredictable and consequently require a high level of real-time adaptability.

From the standpoint of software engineering, this problem is also interesting since it covers most of the issues of analysis, design, implementation, experimentation and validation of artificial organizations in a strictly controlled while moderately abstracted world (Kitano et al., 1995).

The design process (see (Drogoul and Collinot, 1998) for more details) led us to define four individual roles: *shoot the ball, place oneself, block an opponent,* and *defend.* The typology of the relational roles (and thus the dependencies) included the abilities of *coordination, sequential facilitation, simultaneous facilitation* and *inhibition* between two roles. We kept the basic typology of organizational roles (i.e. *initiator* and *participant*). The dependencies layer (and the associated relational roles) provided us with two potential groups centered around the defenders and the shooter. An example of group formation and management is shown in figure 6.

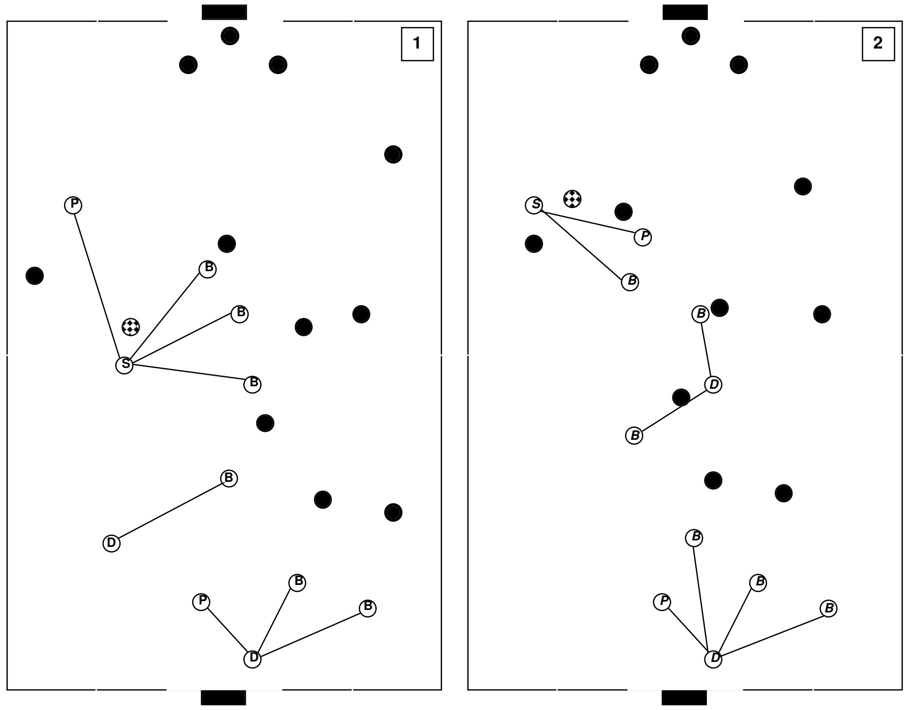

Figure 6 - Example taken from the SoccerServer, showing the evolution of three groups.

Although the use of Cassiopeia has greatly eased the design process of these teams, this work has allowed us to point out the most problematic parts of a methodological approach to the design of MAS, namely the subjective choices the designer has to make at each of the different steps:

1. *Choosing the individual roles* - This choice has a considerable impact on the other steps but nothing prevents the designer from making irrelevant choices. It is clearly an analysis step, but the difficulty is that it takes both the elementary behaviors and the organization to be obtained into account. Moreover, the roles should be fully functional in a single-agent mode as well as in multi-agent modes. The only solution so far is to rely on an informal analysis of the domain, while experimenting groupings of elementary behaviors or functions.
2. *Selecting interesting and consistent dependencies between the chosen individual roles* - Once again, nothing really prevents the designer from making irrelevant choices at this level, since every dependency can be considered as relevant *per se*. Conversely, it is sometimes difficult to foresee the interest of a given dependency, although it can be useful for forming potential groups. The analysis of various sets of experiments and an important knowledge of the domain are usually required for this step.
3. *Choosing the relational roles and the semantics associated with them* - Although some typologies have begun to be proposed (see, for example, the theory of social influences by Castelfranchi (Castelfranchi et al., 1992)), this point is still fuzzy. On the one hand, providing the agents with various relational roles allows them to enter into rich interactions, and can be useful for detailing the dependencies. On the other hand, this makes the agents more complex and thus more difficult to design. A good balance must be kept between richness and simplicity, and the semantics of the roles should be clearly established. Experience shows that it is sometimes difficult to do it without a good knowledge of the DAI techniques for handling the interactions and communications between agents.
4. *Building the potential groups based upon the dependencies* - Simple rules can be applied to the dependencies layer for arbitrarily determining the potential groups. However, when it

comes to choosing which groups will be instantiated, the agents must be provided with specific knowledge about their relative interest .

5. *Choosing the organizational roles and their semantics* - Solid knowledge on the available cooperation and negotiation algorithms and protocols is usually required at this step (at least, some classical ones like the Contract Net or Sycara's negotiation techniques (Smith and Davis, 1980; Sycara, 1989)).

In the above points, given the relatively important amount of ad hoc, non-formalized, subjective or fuzzy information that has to be injected into the methodological process, the question of the real interest of using a design methodology can be asked. The answer to such criticism is usually to point out that the methodology forces this knowledge to be made explicit throughout the different steps. While it is not totally exact (i.e. the definition of the individual roles is a hidden process in *Cassiopeia*), it is certainly better than having no framework for making these choices. However, to be more in the AI way of looking at things, an answer to this knowledge acquisition problem would be to consider part of this information as specific knowledge that can be acquired by the agents for improving their individual or collective behaviors through experience, instead of relying exclusively on the designer's skills.

Precisely, this latter approach amounts to introducing machine learning as one of the key abilities of the designed MAS. In section 3 we review the most promising work that applies machine learning in distributed AI to build adaptive MAS. In section 4, we present *Andromeda,* a full integration of distributed machine learning within *Cassiopeia*. *Andromeda* is meant to assist the designer at each *Cassiopeia* step in providing the agents with learning abilities that allow them to learn all or part of the necessary knowledge (roles, dependencies, group structures, etc.) from experience.

3. MACHINE LEARNING AND MAS-DESIGN - A NECESSARY MARRIAGE FOR SUPPORTING THE DEVELOPMENT OF ADAPTIVE-MAS

3.1. Distributed Machine Learning (DML)

In spite of the reticence of part of the Machine Learning community who predicts that DML is today too hard a problem because Machine Learning is already so difficult in practice, pathfinders have not been stopped from studying the numerous and exciting challenges that learning in a Distributed Environment raises for AI (Dorigo and Schnepf, 1993; Grefenstette et al., 1990; Lin, 1992; Sen, 1997; Sian, 1991 ; Singh, 1992; Stone and Veloso, 1996b; Weiss, 1996 ; Weiss and Sen, 1995). This section supports their view and presents DML as being as inevitable in DAI as ML was for single-agent AI.

3.2. General DML issues

Classically, in machine learning a single agent that learns in an (unknown) environment is considered. Any other agent that co-exists in the same environment is usually considered as part of the environment and therefore not modeled explicitly as such (see figure 7 below). This kind of learning context is called here **single-agent** learning (Dietterich, 1990; Langley, 1996; Mitchell, 1997). In this context, *an agent is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance (as a single-agent) at tasks in T, as measured by P, improves with experience E.*

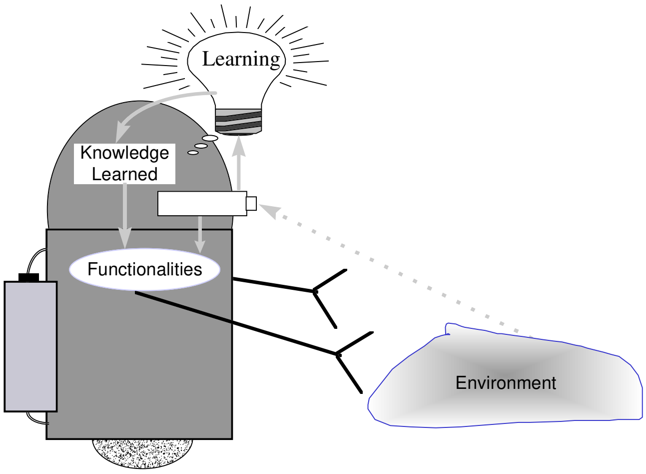

Figure 7: The single-agent machine learning framework

Many techniques have been developed to address the general issue of single-agent learning (Mitchell, 1997). Here is a list of several learning methods in order of the degree of inductive leap required and the degree of supervision they imply:

- Rote learning, which consists in acquiring knowledge and skills that do not require modification from the learner to be applicable (Rachlin et al., 1994). This kind of learning may be characterized by the absence of **generalization**. The knowledge memorized cannot be re-used in new situations.

- Learning from instances (Instance Based Learning (Aha et al., 1991)), which consists in acquiring knowledge and skills that require modifications from the learner only when a new context is encountered and w.r.t. to this context. This learning may be characterized by a form of **local generalization, and** is also referred to as lazy learning in that instances are memorized and the learning *per se* is only performed when necessary and in the context of the task to be solved.

- Learning from examples (concept learning (Dietterich, 1990)), which consists in acquiring knowledge to characterize or discriminate a concept. Such learning may be characterized by the **global generalization** it requires. Initial examples are forgotten and general knowledge that somehow compresses the examples (Quinlan and Rivest, 1989) is stored as decision trees, sets of rules, neural networks, classifiers, etc.

- Learning to build classification (unsupervised learning (Cheeseman et al., 1988; Gennari et al., 1989)), which consists in organizing knowledge. This kind of learning may be characterized by the **absence of supervision**. Classes and their descriptions are built solely from object description.

- Learning by practice or experience (reinforcement learning (Kaelbling et al., 1996), genetic algorithms (Goldberg, 1989)), which consists in acquiring skills from an unknown environment. This kind of learning may be characterized by a form of indirect supervision, i.e. the **reinforcement given by the environment** used to learn (either as a "fitness" in genetic algorithms or "rewards" in reinforcement learning).

The first difficulty raised by the problem of integrating machine learning in a distributed environment is the different levels at which learning may occur (see figure 8 below):

- At the level of an **agent**s, it may learn to achieve its own goals better. At this level, learning is based on system-internal estimates of the **usefulness of the actions carried out by the agent**. Such an agent is called self-interested.

– At the level of **groups of agents**, the group may learn to achieve the goals of the group better. At this level earning is based on system-internal estimates of the **usefulness of the sum of actions carried out by a group** of agents (either homogeneous or heterogeneous).

– At the level of the whole **social organization**, the organization may learn to achieve the goals of the whole organization better. The distinction between groups of agents and the whole organization is meant to differentiate an intermediate level between the agent and the global organization, be it subgroup organization or a one-to-one interrelation. In the *Cassiopeia* methods, a fundamental hypothesis is to consider in particular the different one-to-one agent interactions. At this level, learning is based on the development of system-internal organizational structures imposed on the system components, evaluated on the **usefulness of all the agents taken as a whole** (Weiss and Sen, 1995).

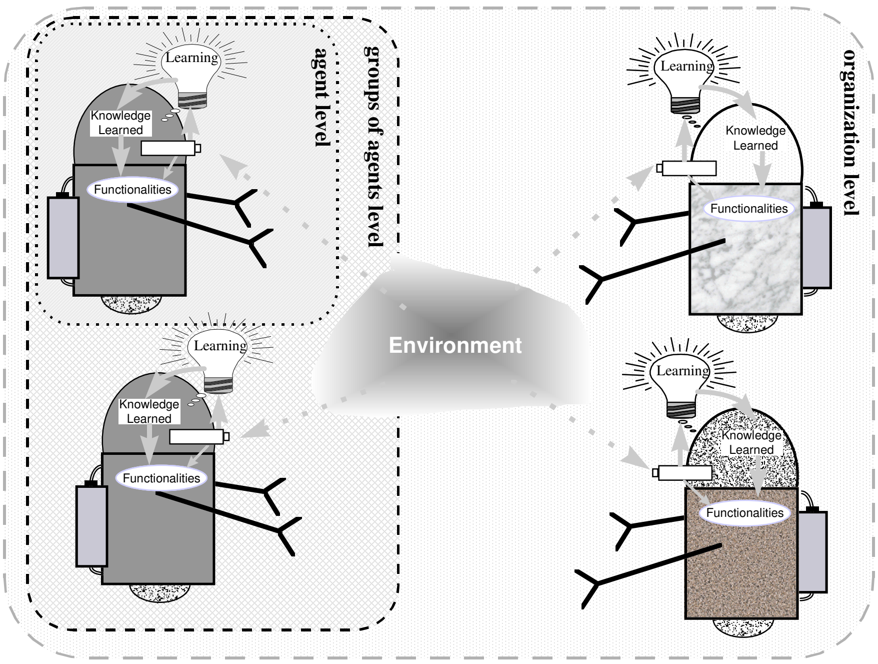

Figure 8 - The multi-agent learning levels

A second serious difficulty is to analyze the learning process itself. It raises several key questions. What is the learning task? what are the learning context and the learning protocol (agents, communication, environment, etc.) ? What is the performance measurement of the learning? These questions receive different answers depending on the learning context, task and learning level: the agent, the group or the whole system. Nevertheless, in practice the different changes that occur in an MAS are due to changes at the level of the agent. The following part of this section proposes a kind of structural analysis of all the possible DML within a DAI environment. The aim is to classify the learning independently of the implementation. Some tasks have already been studied in real applications, others have not yet been studied in depth in DAI.

3.3. Different learning tasks

We propose to differentiate the learning tasks depending on the **beneficiary** of the learning: an agent itself, a group of agents or the whole organization. We do not make this distinction based on the **learner** because in a distributed environment the effective changes are in practice always done at the level of the agent itself. Moreover, such a perspective on

DML allows us to define distributed machine learning in a fashion similar that of the single-agent learning: *an organization (be it a single agent or a group of agents) is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance (as an organization) at tasks in T, as measured by P, improves with experience E*.

This definition has the advantage of being independent of the MAS implementation. Let us note that some researchers have proposed to differentiate the learning tasks based on a different criteria, namely the learning protocols used (Stone, 1996). Next section details the differences between the learning tasks depending on the **beneficiary** of the learning.

3.3.1. Learning basic actions, skills and roles

The first kind of learning tasks characterize situations where an individual agent improves its performances at reaching its goals. At this level, a useful distinction may be drawn between **basic actions**, **skills** and **roles**. We call basic actions those that are considered as primitives within the language used to describe individual roles (IDL). Individual skills are basic behavior that may be described using the IDL.

Learning basic actions is possible and the typical learning performance measurements used in classical ML, i.e. learning rate or success rate over time, are used. Beside the DAI contexts, this kind of learning may be implemented using a large variety of machine learning algorithms (Langley, 1996). In the case of the RoboCup Challenge (Kitano et al., 1997b; Kitano et al., 1997c), the agent learning language includes the actions KICK, DASH, SAY, SEE, etc. Learning to "recognize that the ball is at a kicking distance" (Asada et al., 1996), "discriminate adversaries from partners " are examples of learning tasks at the level of basic actions. Nevertheless, many researchers consider that the set of basic actions (such as running or kicking) is often given (by the hardware configuration corresponding to the simulation considered for example) and thus new *basic* actions ought not to be learnt (Stone and Veloso, 1996b ; Stone and Veloso, 1997a).

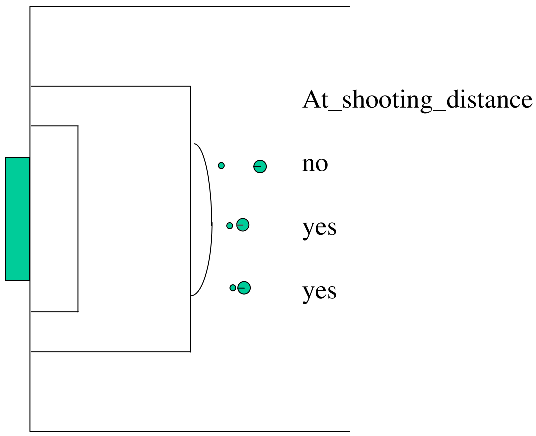

Figure 9 - Three learning examples for the function "is at shooting distance"

In contrast, **individual skills** require that an action language be given. The behavior of the agent will consist in selecting, planning those actions that lead to an individual goal. As for individual skills, **explicit definitions of their individual goals** are required for measuring learning performance. An important choice for an agent within an adaptive multi-agent system is whether to learn directly the expected utility of actions in a given state or to model the other agents in the system. Following Parkes and Ungar (Parkes and Ungar, 1997) we shall call the first type **direct-learning** and the second one **model-based learning**. The standard direct-learning approach is based on Q-learning (Kaelbling et al., 1996). Within model-based learning agents two types of learners may be considered: **myopic-learning** agents that use simple short-term learning models and **strategic-learning** agents that consider the long-term equilibrium of the system. This latter form of learning involves often repeated learning situations over time. Model-based learning requires more complex knowledge representations than those used in reinforcement learning. Model-based learning is typically studied in adversary conditions such as in games (Carmel and Markovitch, 1996; Meyer et al., 1997 ; Stone and Veloso, 1996b). Mataric draws a somewhat similar type of distinctions to classify individual agent strategies: reactive, behavior-based, planner-based and hybrid approaches (Mataric, 1995).

In the case of the RoboCup Challenge, learning to "pass a ball better" is a typical direct-learning task (Stone and Veloso, 1996a). ShooBot (Mizuno&Kourogi&Muraoka, 96) is a real robot that is able to learn to shoot and dribble. It handles numerous parameters (robot speed, ball speed, etc.) to perform its learning (Mizuno&al, 96). Matsubara, Noda and Kazuo consider the task of " shooting in a ball in movement (passed by a partner)"(Matsubara&Noda&Hiraki, 96). They train a neural network to learn this task (Itsuki&Hitoshi&Kazuo, 96). Another key skill for a soccer agent is to be able to intercept the ball (see Fig. 5). This skill is obviously more difficult than going towards an immobile ball (Stone&Veloso, 96b). Intercepting a ball is required from defenders (to oppose a kick or receive a pass), by midfields and attackers (to receive a pass or intercept one from the opponent) Stone&Veloso, 96c) (Stone&Veloso, 95a) (Stone&Veloso, 95c). Stone and Veloso have proposed a Neural-Network-based approach to learn to move towards a ball. Based on the SAGACE approach to anticipation (Meyer et al., 1997), we have developed a **strategic-learning** task consisting in **anticipating** when an attacker will shoot. This latter learning task requires building a model of the adversary decision on when to shoot, which characterizes this kind of strategic learner.

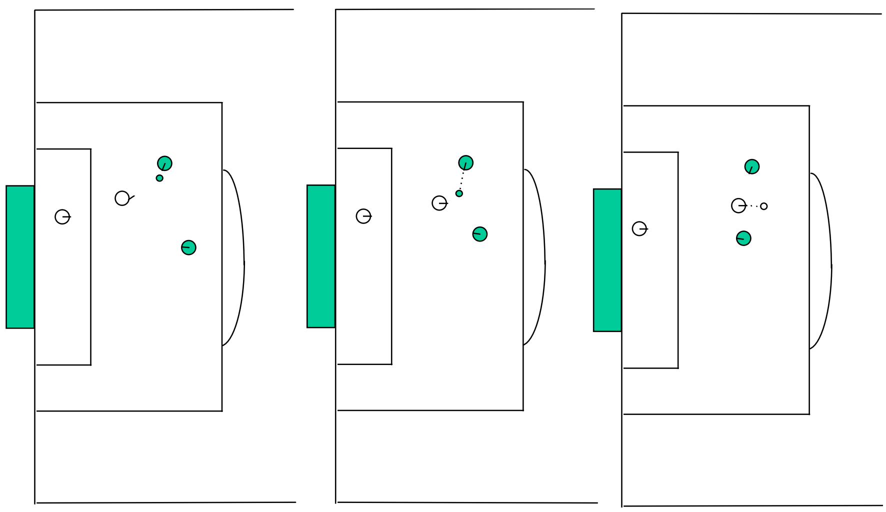

Figure 10 - Learning example of a direct learning task of intercepting a ball

3.3.2. Learning at group level

At the level of a **group of agents**, learning is based on system-internal estimates of the **usefulness of the set of actions carried out by the group** (either homogeneous or heterogeneous). The major problem faced in such learning is the **credit assignment problem**. In single-agent learning, this problem also exists in the reinforcement learning framework. In this case, the learner is not informed of the correctness of actions taken individually but is only informed of their correctness as a whole. The single-agent learner faces the problem of determining the degree to which each action in the sequence deserves credit or blame for the final outcome.

In the case of group-level learning the complexity of the problem is further increased, because it requires determining which set of actions performed by the different agents in the sequence deserves credit or blame for the final outcome. One approach consists in distributing the credit between the different agents (Gu and Maddox, 1995). The problem thus comes down to the single-agent learning being given its share of the global credit. This credit assignment problem has been studied in the framework of classifier systems and algorithms such as the bucket brigade (Dorigo and Bersini, 1994; Goldberg, 1989; Grefenstette, 1988; Grefenstette et al., 1990) are used to solve the multi-agent credit assignment problem (Grefenstette, 1988; Riolo, 1987a; Riolo, 1987b; Wersterdale, 1987).There is an abundant literature on the use of Reinforcement Learning (RL) in distributed environments and the different ways of providing feedback to agents on their actions. It can be done by global systems or other agents (inter-agent credit-assignment (Goldman,97)), Mataric, for example, propose to define *progress estimators* so as to help the agent better evaluate its progression towards a goal (Mataric, 1994a).

In the case of the RoboCup Challenge, "to score more goals" is an example of task for a set of attackers. An obvious credit may be the number of goals scored per minute. The assignment of the credit may then be shared amongst the attackers. To simplify, a heuristics may be to give most of the credit to the striker and the rest equitably between the other attackers. Other heuristics may give more emphasis to the sequence of actions and give more credit to the player that passed the ball to the striker (Stone and Veloso, 1996b; Stone et al., 1996). Many other group-level learning tasks may be represented by the ability of individual agents to select the skills "when to dribble, pass or shoot", " anticipate positioning before receiving ball from partner", "block the opponent", "set a trap to intercept a ball". Aubineau & Lalande (1996) have developed an approach based on Reinforcement Learning to learn to select a *Cassiopeia* individual role, that is to learn relational roles.

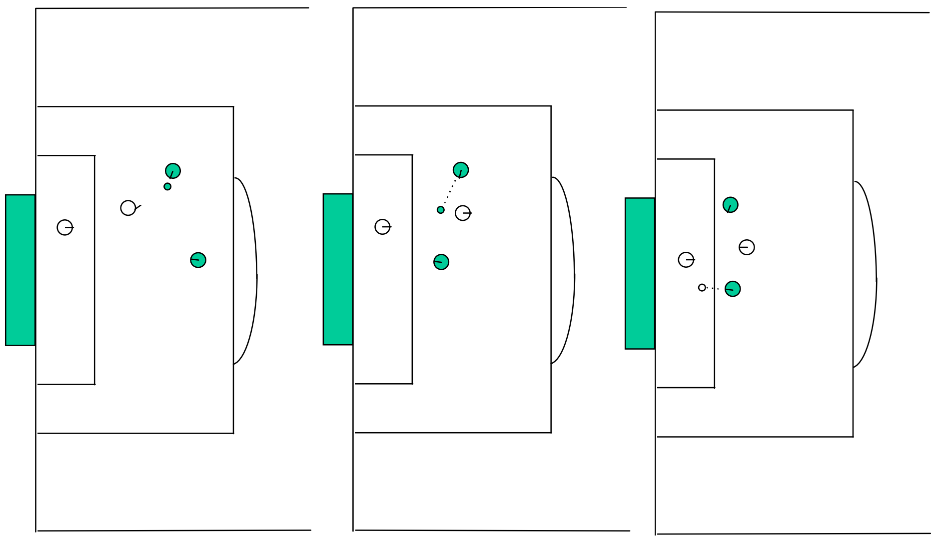

Figure 11 - A group learning a "one-two pass" task

3.3.3. Learning at the organization level

At the level of the whole **social organization**, learning is based on the development of system-internal organizational structures imposed on the system components evaluated on the **usefulness of the whole group.** (Ito and Yano, 1995; Lund, 1995; Mataric, 1994b). A promising approach used to make a social organization learn is the use of genetic programming (GP), a class of adaptive algorithms used to evolve solution structures that optimize a given evaluation criterion. The social evolution of strategies may be investigated using genetic algorithm techniques (Takadama and Nakasuka, 1996). Ito shows for example that robust and cooperative strategies emerge through out the evolution starting from simple ones (Ito and Yano, 1995). Haynes et al., represent cooperation strategies that can be manipulated by GPs (Haynes et al., 1995b). Their results in the predator-prey domain show that with minimal input of domain knowledge and human intervention, the construction of good cooperation strategies can be built. In the case of the RoboCup Challenge, many researchers aim at such a level of learning (Stone et al., 1996) but few results have been obtained so far. "Increase the occupation of the adversary field", "increase the time the team has kept the ball", are typical and difficult learning tasks at the organization level.

Figure 12 below summarizes different forms of DML reviewed in the previous section.

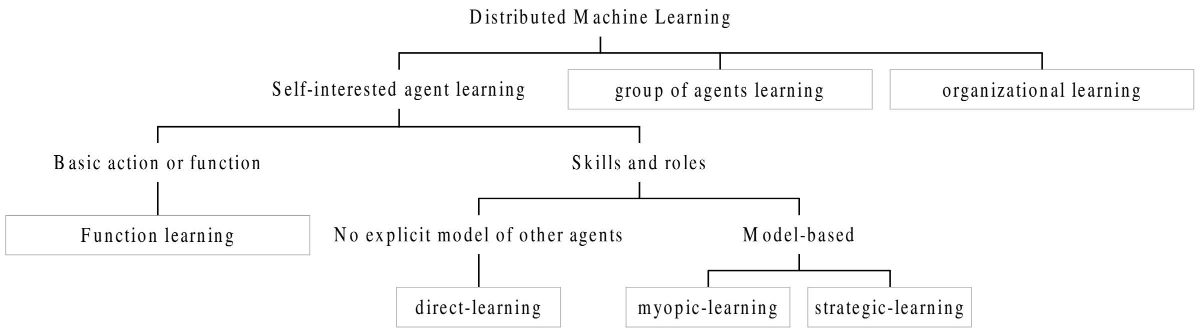

Figure 12 - Different forms of DML

3.4. Different learning protocols

3.4.1. The training experience

Whoever is learning, the richness of MAS allows many protocols to be defined with different environments, different sets of agents (homogeneous or heterogeneous), different agent-to-agent interactions and agent-environment interactions (Sen, 1997; Weiss, 1996). The choice of protocol is directly linked to the nature of the training experiences from which the multi-agent system will learn. The different types of training experiences can have a significant impact on the success or failure of the learning. In a DAI environment, there are many properties of the protocols that restrict the type of learning possible. One issue is whether the feedback is direct or indirect w.r.t. to the task to be learned. We discussed above the importance of this issue for credit assignment. A second issue is the context in which the agent learns, with one or many other agents, with a learning agent or not, in repeated conditions, in real conditions, etc. A third issue is related to the learnability of the task considering the training experience that is at hand. This latter issue has been studied extensively in single-agent learning (Kearns, 1990) but this area is still open in multi-agent learning.

In the case of the RoboCup Challenge, many protocols may be conceived to build training experiences for learning. As an example, let us consider the setting used by Stone and Veloso to train a Neural Network-based defender that ought to learn the angle it should turn to stop the ball. Stone and Veloso propose to consider a set of experiences defined using the following movement generator where BD = Ball Distance, BA = Ball Angle, TA = Turn Angle:

```
while BD > 14, TURN (BA)
when BD <= 14, set TA = Random Angle between -45 and 45
record BD, BA, previous BD, TA
TURN (BA + TA)
DASH
record result (from coach)
```

As long as the ball is further than a given threshold, the defender observes the ball. When it becomes close enough, the defender chooses a random angle and moves. A coach is used to label such produced examples as GOAL when the player scores, SAVE when the ball is recovered, and MISS in other cases. These generated examples are then used to train a neural network to learn the proper angle the defender should turn in order to stops goals. Stone and Veloso have proposed for each characteristic to learn to isolate situations requiring such characteristics and to produce examples from these situations.

In the case of the RoboCup Challenge, the learning protocols may be broken down into several types (Kitano et al., 1997c) including: off-line skill learning by individual agent, off-line collaborative learning by teams of agents, on-line skill and collaborative learning (the ability to adaptively change collaborative or individual behaviors during the course of a game could contribute to a team's success) and on-line adversarial learning, i.e. learning to react to predicted opponent actions.

3.4.2. Game-theory based protocols

The most characteristic protocols in DML are the ones that include more than one learning agent. Game theory has emerged as one of the most interesting frameworks for studying multi-agent learning protocols (Fudenberg and D., 1998). According to Parkes and Ungar (Parkes and Ungar, 1997)"m*ulti-agent systems can be viewed as games where the artificial agents are bounded-rational utility-maximizers with incomplete information about the other agents in the systems*". The problem of the designers of MAS may then be viewed as one of establishing rules of the game that encourage agents to choose strategies that optimize system-wide utility. Game theory offers elaborate studies of the effect of deserters on individual optimality (Axelrod 1984), but the domains it treats are typically much more clearly constrained than environments in which robots are situated. In particular, game theory deals with rational agents capable of evaluating the utility of their actions and strategies ; a setting that is seldom found in MAS. To cope with such contexts in DML, adversarial behaviors may be **inferred** from the adversaries' past behavior (Carmel and Markovitch, 1996; Meyer et al., 1997 ; Mor et al., 1996). The progress over the last few years in the design of adaptive learning algorithms, with the help of genetic algorithms, has made it possible to make such inferences thanks to which the machines build for themselves an internal representation of their adversaries.

3.5. An immense unexplored field with as yet promising but scarce results

We present below a brief overview of the techniques used in single-agent learning that have been adapted to DML. In DML the most studied learning problem can be formulated as learning state-action or state-action-value functions that represent the best action that an agent should take given a past history of rewards and actions of the other agents (Parkes & Ungar, 97). The state of the world models all of the information that an agent uses to adapt its behavior. The main approach explored so far to learning in distributed systems is **reinforcement learning** (Stone, 1996; Weiss and Sen, 1995). The principle is straightforward: the tendency to choose an action in a given state is strengthened if the result it produces is favorable, weakened otherwise. Many other learning techniques can also be implemented.

Machine Learning Techniques	Machine Learning Task in DAI
Reinforcement Learning (Asada et al., 1996; Gu and Maddox, 1995; Lin, 1992; Schwartz, 1993)	**All the learning tasks** – Improving basic actions – Improving individual skills – Improving individual roles (single-agent, direct, myopic) – Improving group behavior (coordination) – Improving social behavior (cooperation) – Improving communication
Neural Networks (Parisi et al., 1994; Stone and Veloso, 1996a; Werner, 1994)	**Numerical classification tasks** – Improving basic actions – Improving group/behavior (coordination)
Case-Based Reasoning (Haynes et al., 1996; Plaza et al., 1995)	**Knowledge-sharing tasks** – Sharing distributed symbolic knowledge
Instance-Based Learning (Decker and Lesser, 1995; Haynes, 1997; Nagendra Prasad and Lesser, 1997)	**Decision/classification tasks** – Improving group team behavior (coordination)
Inductive Learning (Davies and P. Edwards, 1996; Goldman and Rosenschein, 1995 ; Potter et al., 1995; Stone and Veloso, 1997b)	**Symbolic classification and knowledge sharing tasks** – distributed classification – strategic learning – Sharing /confronting distributed symbolic knowledge – deciding to be altruistic
Genetic Algorithms (Fogarty et al., 1995; Schultz, 1994) **Genetic Programming** (Haynes et al., 1995a; Ito and Yano, 1995)	**Optimization tasks** – Improving individual role (single-agent, direct, myopic) – Improving group behavior (coordination) – Improving social behavior

Table 3-1 - Overview of the different ML techniques that can be applied to DAI

3.6. No fully-integrated methodologies

The examples of DML tasks that have been presented above have shown in many cases the benefit of integrating DML in an MAS. The problem of integrating ML in the design methodology has not been studied much. The first benefit of integrating DML in the design process is to produce final systems that are adaptive and learn to improve their global performance. There is a consensus in the AI community that such a goal is per se a desirable one. The second benefit is at the level of the methodology itself. We have pinpointed several difficulties resulting from the use of Cassiopeia. In particular, the design decision may require an expertise that is not necessary when developing an MAS. An integration of DML in the design process would therefore attempt to make such steps easier and at the same time guarantee that the produced architecture is adaptable.

Mataric in a seminal paper has proposed a methodology which relies mainly on one machine learning technique, reinforcement learning. Her approach, called *"basic behavior approach"* (Mataric, 1995) is a "methodology [for] automatically generating higher-level behaviors by having the agents learn through their interactions with the world and with other agents, i.e. through unsupervised reinforcement learning". Although a pioneer in the field, her approach has several flaws: the learning technique used is restricted to reinforcement learning, the design process is a sequential one that does not easily accommodate iterative design. In the next section we propose our approach to the issue of defining a general framework for integrating machine learning techniques.

4. *ANDROMEDA:* A FRAMEWORK AND A METHODOLOGY FOR INTEGRATING MACHINE-LEARNING IN THE MAS DESIGN

4.1. Introduction

To employ DML it is necessary to define the level at which learning is expected, define the learning task and its evaluation and finally to define protocols for learning. We shall see in the following how *Cassiopeia* may be used to organize the different levels where *Andromeda* uses DML. The proposed modification in *Andromeda* is to be able to decompose not in terms of behaviors (which allows an immediate implementation) but in terms of **goals**. This change requires that the designer elicitate the knowledge at the level of **goals** and **not functionalities or behaviors.** This is the **key** of the *Andromeda* approach**: adaptive behaviors or ones that will be learned are defined in terms of their goal and not in terms of a program,** with no other semantics but that of the behavior produced. By doing so *Andromeda* allows the designer to define both the goals as a designer and the material necessary for performing the learning steps. As a consequence, the designed MAS may be adaptive so as to reach the defined goals in evolving environments. The next five sections present the design activity in *Andromeda* corresponding to the *Cassiopeia* layers, i.e. the methodological framework.

4.2. Phase 1 - Learning individual actions, skills and individual roles

In *Andromeda*, phase 1 concerns the definition of individual actions, skills and individual roles. The language used for describing individual roles is called the individual role language (IRL). In this language that will not be described formally here, we distinguish between actions, skills that are composed of basic actions, and individual roles composed of skills. In the RoboCup Challenge, the IRL basic actions are given: *TURN*, *DASH*, *KICK*, *SAY*, *CHANGE VIEW*, *MOVE*. Any particular programming language using these actions as primitives may be taken for an IRL.

In this first phase, the beneficiary of the learning is obviously the agent itself. Its task consists in learning its roles in IRL with no explicit modeling of the other agents, but with an explicit description of the goals to be reached and their performance measurements. In that respect, the major difference with *Cassiopeia* is that the individual roles are seen as dynamic planners provided with goals. To reach these goals, the agents can either build new behaviors or tune given behaviors by adjusting their parameters. The table below summarizes the main characteristics of this phase.

Beneficiary	A single agent
Task	– Learning a **behavior** in an individual role language (**IRL)** – with **no explicit** modeling of other agents per se – with an **explicit** description of the goal to be reached
Approaches	Define a self-interested goal G and performance measurement P for: – **building** a behavior represented in IRL for reaching G, or – **tuning** a behavior by adjusting parameters for reaching G
Techniques	Reinforcement learning, Genetic Algorithms, Neural Networks
Difficulties	The expressive power of IRL, the description of the goal G, the learning performance measurements P, the choice of learning protocols.
Protocols	One given agent: 1. alone in the environment (either off-line or on-line) 2. with one to many non adaptive agents (either off-line or on-line) 3. with one to many adaptive agents (either off line or on-line) 4. in real conditions (on-line)

Table 4-1 - *Andromeda's* Phase 1 in a nutshell

From a methodological point of view, the aim of this phase is to assist the designer in the definition of the individual roles by allowing the agents to learn or tune them. But it is not intended to replace the designer's work: there is still, of course, the possibility to define behavior-based roles (as in *Cassiopeia*) along with the new goal-based roles. We have, for example, used this ability in the design of the simulation team that will be competing in RoboCup'98 (Kitano et al., 1997b): while the attackers rely on previously defined reactive behaviors (Drogoul and Duhaut, 1996), the defenders have learned new behaviors. Moreover, "innate" and acquired roles can coexist in the same agents without any problem.

4.3. Phase 2 - learning one-to-one dependencies

Andromeda's phase 2 concerns the building of the one-to-one dependencies between the individual roles. This corresponds to the dependencies layer of *Cassiopeia*. The language used for describing influences is IDL. This language is made up of predicates such as ENABLE, FACILITATE, REDUNDANT, GOAL COMPETITION, RESOURCE COMPETITION, INDEPENDENT, INTERFERENCE, INCOMPATIBLE, COORDINATION. The semantics of the terms in the IDL is given with respect to the impact of other individual roles on the performance of the agent. Given R1 and R2, two agents having individual roles R1 and R2 characterized by performances P1 when R2 is absent and P'1 when R2 is present, here follows the semantics of a few terms we are using in our IDL:

```
(P1=0 and P'1≠0 )   →   ENABLE(R1,R2)
(P1=P'1)            →   INDEPENDANT(R1,R2)
(P'1>P1)            →   FACILITATES(R1,R2)
(P'1=0)             →   INCOMPATIBLE(R1,R2)
```

With respect to the possibility to introduce hand-coded individual roles in phase 1, it has to be noted that dependencies between agents whose individual role is not attached to a performance measurement P need to be given by the designer because their dependencies with other agents cannot be learned.

The beneficiary of the learning is once again a sole agent. Its task consists in learning, in IDL, the influences that the roles of the agents it is interacting with exert on itself, which amounts to building an explicit model of these agents. The table below summarizes the characteristics of this phase:

Beneficiary	An individual agent in a given individual role and associated goal G
Task	Learning the influences of other individual roles on one's own current individual role by **explicit** modeling of the other agents
Approaches	Use the definition of the self-interested goal G for: – **building** a model of the influence of other individual roles – **tuning** a behavior by adjusting parameters for reaching G when confronted to other individual goals
Techniques	Relational learning (Aha, 1992 ; Giordana et al., 1997; Muggleton and Raedt, 1994; Zucker et al., 1998) Concept learning (Dietterich 1990)
Difficulties	The expressive power of IDL, the choice of learning protocols.
Protocols	Two agents with a fixed individual role R1 and R2 where: 1. R1 is adaptive and R2 is not (either off line or online), 2. R1 and R2 adaptive (either off line or online), 3. Real condition (online)

Table 4-2 - *Andromeda's* Phase 2 in a nutshell

The methodological interest of this phase is obvious: only the meaningful dependencies will be kept, that is the dependencies that express an influence between two agents when they are playing two given individual roles. However, the key role of the protocol used for the learning task should not be underestimated: it relies on the same hypothesis as that being made in *Cassiopeia*, namely that the definition of one-to-one dependencies is sufficient to constitute relevant groups. In the cases where an agent simultaneously needs to influence or be influenced by a number of agents that play the same role, it is the duty of the designer to apply this knowledge and define specific protocols. Working under this hypothesis nevertheless means that it simplifies performance evaluation during the learning process.

4.4. Phase 3 - learning relational roles

In *Andromeda*, phase 3 concerns the building of the relational roles. The language used for describing relational roles is RRL. This language is made up of predicates such as INFLUENCING and INFLUENCED. which correspond to the typology of relational roles introduced in Cassiopeia. The major difference with *Cassiopeia,* however, is that the relational roles are defined using goals which allow flexible relational roles that may be learned in different contexts.

The relational roles are built primarily by analyzing the dependencies learned in terms of "active" influences. The fact that the role played by an agent A dependis on another role (i.e. that must be played by another agent) should both enable A to influence the other agents in a specific way, and induce these agents to react accordingly (i.e. by undertaking the corresponding role). The goals to be reached when trying to learn the relational roles are therefore threefold: (1) knowing which influence sign to produce and when to produce it; (2) knowing how to choose among different influence signs and (3) knowing how to react to an influence sign by selecting the appropriate individual role. The actions or skills that will be undertaken when triggering these roles can either be provided by the designer (i.e. specific communication protocols or role selection mechanisms) or, as in the case of the individual roles, learned by the agents.

In this case, the beneficiary of the learning task is not one agent, but a set of two agents that in fact learn to coordinate themselves with respect to their dependencies. The table below summarizes the characteristics of this phase:

Beneficiary	Two agents having several individual roles
Task	Learning coordination and relational roles in **RRL** with **explicit** modeling of the other agent's individual roles
Approaches	– building Model: finding when to change individual role – building Model: finding when informing about one's individual role – tuning a model: adjusting own parameters for changing of individual roles.
Techniques	Relational learning, Genetic algorithms Instance Based Learning, Genetic programming
Difficulties	The expressive power of RRL, Learning measurements
Protocols	Two agents: 1. two adaptive agents (either off-line or on-line) 2. one non-adaptive agent (either off-line or online) 3. in real conditions (online with all the other agents)

Table 4-3 - *Andromeda's* Phase 3 in a nutshell

4.5. Phase 4 - learning to build groups of agents

In *Andromeda*, phase 4 concerns assisting the designer in defining the groups of agents. The language used for group definition is called GDL. As in *Cassiopeia*, a group of agents is built upon the one-to-one dependencies and corresponds to a path on the dependencies graph. Although being provided with relational roles is in most cases sufficient for the agents to **coordinate** themselves, the idea behind establishing these groups is to enable the agents to explicitly **cooperate** by identifying the regularities in the simultaneous activation of their individual roles by their relational roles.

In this phase, the beneficiary of the learning is therefore a set of agents (although the identification is still being made at agent level). The table below summarizes the characteristics of this phase:

Beneficiary	A group of agents already provided with their relational roles
Task	Identifying configurations in **GDL** with **explicit** modeling of the other's individual roles
Approaches	Building groups
Techniques	Conceptual clustering (Stepp and Michalski, 1986) to organize roles Case Based Reasoning (Kolodner, 1993) to memorize previous design case
Difficulties	Evaluation of group usefulness
Protocols	Unsupervised learning protocols

Table 4-4 - *Andromeda's* Phase 4 in a nutshell

The designer can orient the identification of the groups by providing specific goals (and their corresponding performance measurements) that should be collectively reached by the sets of agents. This enables the agents, when two different groups can be formed for the same end, to compare their performance with regard to these goals.

4.6. Phase 5 - organizational role

In *Andromeda*, the fifth and last phase concerns the definition of organizational roles in a language called ORL, which contains predicates similar to the types of roles defined in the typology introduced in the fifth layer of *Cassiopeia*. The organizational roles play the same role w.r.t. the groups as the relational roles w.r.t. the dependencies. For a set of agents, the goal to be reached in this case is not simply to rely on their relational roles for selecting the appropriate individual roles, but to be able to identify the situations in which they can actively and dynamically manage the collective selection of a given set of roles previously identified as a group.

The behaviors or skills that will be undertaken by the agents when triggering one of these roles can either be supplied by the designer (i.e. specific negotiation protocols or cooperation schemes) or learned by the agents. The first option is nevertheless likely to be the most common case since the latter is a very difficult learning task.

The beneficiary of the learning is theoretically the entire set of groups of agents, i.e. the overall MAS. However, tuning the global behavior of the MAS means that the designer must be able to define one or more global utility functions, which is not always the case. So, in practice, the beneficiaries are the group of agents for which such a function can be easily defined.

Beneficiary	All the groups of agents
Task	**Explicit** modeling of the dynamics of groups in the ORL.
Approaches	- building Model of others: finding when enabling/disabling the relational roles - tuning a model: adjusting own parameters for changing the relational roles.
Techniques	Relational learning, Genetic algorithms Instance Based Learning, Genetic programming
Difficulties	Conceiving organizational roles
Protocols	Real conditions, where performances are measured by the global utility function of the MAS.

Table 4-5 - *Andromeda's* Phase 5 in a nutshell

4.7. *Andromeda* methodological process

Andromeda's basic principle is to let the learning experiments take place between each of the layers definitions in *Cassiopeia*. In that way, for example, once the designer has defined the goals to reach within the first layer, the corresponding roles are learnt by the agents (in an environment depending on the protocol chosen) and then reintroduced as individual roles in the same layer (possibly besides roles already hand-coded by the designer). The goal of the methodology is to keep a maximum of flexibility in the design process, that is to allow the designer to introduce knowledge at the different levels, while supporting the possibility to be assisted by the learning system when the overall complexity becomes too important, or when he does not really manage all the subtleties of the domain.

Keeping a good balance between these hand-coded and learnt roles should allow the designer to keep on managing the design process, while allowing the system to propose him new original solutions. These two approaches can be combined very freely, the designer may:

- entirely design a given layer and rely on the agents for learning the subsequent ones. We have for instance used this approach for learning the relational and organizational roles of a team of robots whose individual roles had been previously defined.
- partially define the different roles, requiring from the system that it completes them with respect to certain goals, which authorizes a very powerful incremental approach.
- design the whole system and rely on the agents for tuning the choices made, by adjusting the parameters used at each level.

The design phases order, as in *Cassiopeia,* is not really prescribed and allows to undertake an incremental as well as iterative design process. However, it is a little bit more difficult to start at any of the phases, since they usually require that the previous ones be either partially designed or learnt. We therefore recommend a bottom-up process, in which the order of the steps corresponds to the numbering of the phases. The figure below presents the integration the *Cassiopeia's* layers and *Andromeda's* phases.

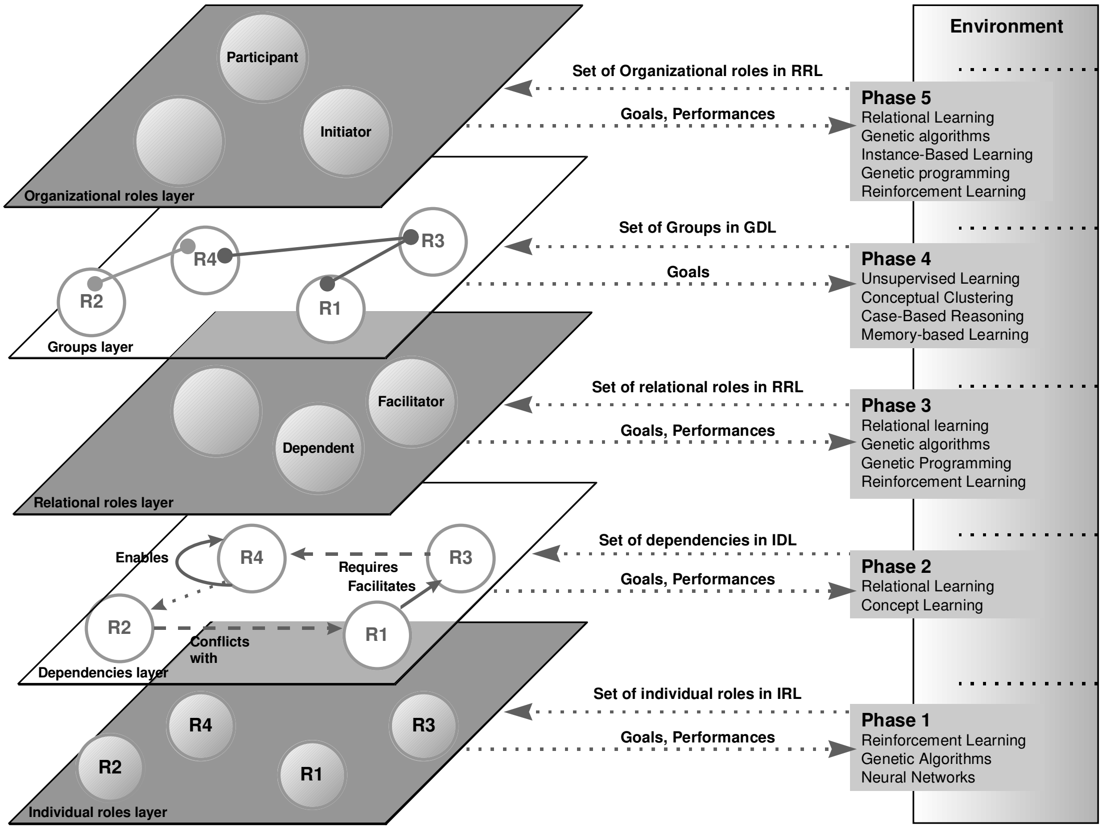

Figure 13 - The five layers of *Cassiopeia* and the corresponding learning phases used in *Andromeda*

The major difference with the *Cassiopeia* design process is that the roles are defined using goals which afford to have flexible roles that may be learned in different contexts. The learning approaches may vary from:

- **knowledge intensive** approaches: the behavior is defined by the designer as a parameterized description in the corresponding language and learning consists in adjusting the parameters depending on the environments. When no parameters exist, this phase in *Andromeda* is similar to that of *Cassiopeia*.
- **learning intensive** approaches: the behavior defined in terms of its goal may be learned in the corresponding language (IRL, IDL, GDL or ORL) from experiences.

5. CONCLUSION

One of the most challenging and, in our opinion, little addressed questions that can be found in Distributed Artificial Intelligence today is that of the methodological design of a learning multi-agent system. In this paper we have shown how an agent-oriented, role-based, method, *Cassiopeia,* could be used to classify and integrate Machine Learning (ML) techniques in the design process of an MAS by carefully considering the different levels of behaviors to which they can be applied and the techniques which appear to be best suited at each level. Using the RoboCup Challenge, we have shown the large part of the MAS-design activity often left to be done as a result of heuristic choices or experimental activities.

One key motivation for integrating machine learning techniques into the MAS design process is to develop adaptive MAS. It has appeared that such integration allows both the development of adaptive agents and also supports the development process itself. The more than fifty years of experienced accumulated in the field of machine learning (Saitta and Neri, 1998) have shown that learning is not the sole means to acquire knowledge and that knowledge-intensive approaches have been successful. For this reason, we consider that *Andromeda* forms a bridge between knowledge-intensive methods such as *Cassiopeia* and more learning-intensive approaches that rely on machine learning. In agents designed using the *Andromeda* methodology, there ought to be a good balance between their learnt behaviors and innate skills.

Finally, we consider that *Andromeda* is the first attempt to integrate the *variety* of machine learning techniques to support the MAS-design process and also design learning MAS. Our current work is focused on the development of an integrated platform to support the development of MAS following the *Andromeda* approach.

On a theoretical level, we are working on the definition of a strong formalization of the five different description languages used at the various architectural level that, besides being used in the various learning tasks, will serve in *Cassiopeia* itself for describing the different layers. On a practical level, we have begun using Andromeda for designing the four teams of soccer-playing robots[3] we are currently building with the support of the CNRS and the University of Paris 6. This should soon provide us with a rich base of experimental data.

[3] That should enter the four competitions defined in RoboCup : simulation league, small-size, middle-size and legged robots.

6. REFERENCES

Abbott, R. J. 1983. *Program Design by Informal English Sentences*. *Comm. ACM*, 23(11): 882-894.

Aha, D. W. 1992. Relating Relational Learning Algorithms. *Inductive Logic Programming*, ed., Muggleton, S., pp. 233-259, Harcourt Brace Jovanovich, London.

Aha, D. W., Kibler, D., and Albert, M. 1991. Instance-based learning algorithms. *Machine Learning*, 6: 37-66.

Asada, N., Noda, S., Tawararatsumida, S., and Hosoda, K. 1996. Purposive Behavior Acquisition for a Real Robot by Vision-based Reinforcement Learning. *Machine Learning*, 23.

Bond, A. H., and Gasser, L. 1988. Readings In Distributed Artificial Intelligence. , Morgan Kaufman.

Booch, G. 1994. *Object-Oriented Analysis and Design*, Benjamin Cummings.

Carmel, D., and Markovitch, S. 1996. Opponent Modeling in Multi-Agent Systems. *Distributed Artificial Intelligence Meets Machine Learning: Learning in multi-agent environments*, ed., Weiss, G., pp. 40-52, Springer-Verlag.

Castelfranchi, C., Miceli, M., and Cesta, A. 1992. *Dependence relations among autonomous agents*. *Decentralized A.I. 3*, ed., Werner, E. and Demazeau, Y., North Holland.

Cheeseman, P., Self, M., Kelly, J., and Stutz, J. 1988. Bayesian Classification. Proc. *Seventh National Conference on Artificial Intelligence*, St. Paul, Minnesota.

Coad, P., and Yourdon, E. 1991. *Object-Oriented Analysis*, Yourdon Press, Englewood Cliffs, New Jersey.

Collinot, A., and Drogoul, A. 1998. *Using the Cassiopeia method to design a Soccer Robot Team*. *Applied Artificial Intelligence (to appear).*

Davies, W., and P. Edwards. 1996. The Communication of Inductive Inferences. *Adaptation and Learning in Multiagent Systems*, ed., Weiss, G. and Sen, S., pp. 223-241, Springer Verlag, Berlin.

Decker, K., and Lesser, V. 1995. Designing a family of coordination algorithms. Proc. *ICMAS*, pp. 73-80, June 1995, San Fransisco, USA.

Decker, K. S. 1987. *Distributed Problem-Solving techniques: A Survey*. *IEEE Transactions on Systems, Man and Cybernetics*, 17(5).

Demazeau, Y., and Muller, J.-P. 1991. Decentralized A.I. 2. , North Holland.

Dietterich, T. 1990. Inductive Learning from Preclassified Training Examples. *Readings in Machine Learning*, ed., pp. 45-56.

Dorigo, M., and Bersini, H. 1994. A Comparison of Q-Learning and Classifier Systems. *Proceedings of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior*, ed., Cliff, D., Husband, P., Meyer, J. A., and Wilson, S. W., Bradford / MIT Press, Cambridge, MA.

Dorigo, M., and Schnepf, U. 1993. Genetics-based Machine Learning and Behaviour-based Robotics : A New Synthesis. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(23): 141-154.

Drogoul, A., and Collinot, A. 1998. Applying an Agent-Oriented Methodology to the Design of Artificial Organisations : a Case Study in Robotic Soccer. *Autonomous Agents and Multi-Agent Systems*, 1(1).

Drogoul, A., and Dubreuil, C. 1991. Eco-Problem-Solving Model: Results of the N-Puzzle. *Decentralized A.I. 3*, ed., Werner, E. and Demazeau, Y., pp. 283-295, Elsevier North Holland, Amsterdam.

Drogoul, A., and Duhaut, D. 1996. *MICROB: Making Intelligent Collective ROBotics*. Proc. *MiroSot'96*, Taejon, Korea.

Drogoul, A., and Ferber, J. 1993. From Tom-Thumb to the Dockers: Some Experiments with Foraging Robots. *From Animals to Animats II*, ed., Meyer, J. A. and Wilson, S., pp. 451-459, MIT Press, Cambridge.

Fogarty, T. C., Bull, L., and Carse, B. 1995. Evolving Multi-Agent Systems. *Genetic Algorithms in Engineering and Computer Science*, ed., Winter, G., Périaux, J., Galan, M., and Cuesta, P., Wiley & Sons.

Fudenberg, D., and D., L. 1998. Theory of Learning in Games, MIT Press, forthcoming.

Gennari, J. H., Langley, P., and Fisher, D. 1989. Models of incremental concept formation. *Artificial Intelligence*, 40-1(3): 11-61.

Giordana, A., Neri, F., Saitta, L., and Botta, M. 1997. Integrating multiple learning strategies in first order logics. *Machine Learning*, 27: 209-240.

Goldberg, D. E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, MA.

Goldman, C. V., and Rosenschein, J. S. 1995. Mutually Supervised Learning in Multiagent Systems. *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, ed., Sen, S., pp. 20--25.

Graham, I. 1994. *Object-Oriented Methods*, Addison-Wesley.

Grefenstette, J. J. 1988. Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms. *Machine Learning*, 3(2/3): 225--245.

Grefenstette, J. J., Ramsey, C. L., and Schultz, A. C. 1990. Learning Sequential Decision Rules Using Simulation Models and Competition. *Machine Learning*, 5(4): 355--381.

Gu, P., and Maddox, A. B. 1995. A Framework for Distributed Reinforcement Learning. *Adaptation and Learning in Multi-Agent Systems, IJCAI'95 Workshop, LNAI 1042*, ed., Weiss, G. and Sen, S., Springer, Montréal.

Haynes, T. 1997. Collective Memory Search : Exploiting an Information Center for Exploration. *Adaptive Behavior*(Special issue on collective intelligence).

Haynes, T., Lau, K., and Sen, S. 1996. Learning Cases to Compliment Rules for Conflict Resolution in Multiagent Systems. *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, ed., Sen, S., pp. 51--56, Stanford University, CA.

Haynes, T., Sen, S., Schoenefeld, D., and Wainwright, R. 1995a. Evolving Multiagent Coordination Strategies with Genetic Programming. Proc. *Artificial Intelligence*.

Haynes, T., Wainwright, R., and Sen, S. 1995b. Evolving Cooperation Strategies. *Proceedings of the First International Conference on Multi--Agent Systems*, ed., Lesser, V., pp. 450, MIT Press, San Francisco, CA.

Hewitt, C., and Inman, J. 1991. Distributed Artificial Intelligence Betwixt and Between: From "Intelligent Agents" to Open Systems Science. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6): 1409-1419.

Ito, A., and Yano, H. 1995. The Emergence of Cooperation in a Society of Autonomous Agents. *Proceedings of the First International Conference on Multi--Agent Systems*, ed., Lesser, V., pp. 201--208, MIT Press, San Francisco, CA.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. 1996. Reinforcement Learning: A Survey. *Journal o Artificial Inteeligence Research*, 4: 237-285.

Kearns, M. J. 1990. The computational complexity of machine learning, MIT Press, Cambridge, London.

Kitano, H., Asada, M., Osawa, E., Noda, I., Kuniyoshi, Y., and Matsubara, H. 1997a. *RoboCup: A Challenge Problem for AI. AI Magazine*, 18(1).

Kitano, H., Asada, M., Osawa, E., Noda, I., Kuyinochi, Y., and Matsubara, H. 1997b. RoboCup: A Challenge Problem for AI. AI magazine.

Kitano, H., Minoru, A., Kuniyoshi, Y., Noda, I., and Osawa, E. 1995. *RoboCup: The Robot World Cup Initiative*. Proc. *Workshop on Entertainment and AI/Alife, IJCAI*, Montréal, Canada.

Kitano, H., Tambe, M., Stone, P., Veloso, M., Coradeshi, S., Osawa, E., Matsubara, H., Noda, I., and Asada, M. 1997c. The RoboCup Synthetic Agent Challenge 97. Proc. *Fifteenth International Joint Conference on Artificial Intelligence, IJCAI'97*, pp. 24-29, August 23-29, Nagoya, Japan.

Kolodner, J. 1993. Case-Based Reasoning, Morgan Kaufmann, San Mateo, CA.

Korf, R. E. 1992. A Simple Solution to Pursuit Games. Proc. *11th DAI Workshop*, pp. 183-194, Boston.

Langley, P. 1996. Elements of Machine Learning, Morgan Kaufmann.

Lin, L.-J. 1992. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Machine Learning*, 8(3/4): 293--321.

Lund, H. K. 1995. Specialization under Social Conditions in Shared Environments. Proceedings *of the Third European Conference on Artificial Life ECAL'95*.

Mataric, M. 1994a. Reward Functions for Accelerated Learning. Proc. *International Conference on Machine Learning*, pp. 181-189, New-Brunswick.

Mataric, M. J. 1994b. Learning to behave socially. Proceedings *of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior*, Cambridge, MA.

Mataric, M. J. 1995. Learning in Multi-Robot Systems. Proceedings *of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95), Workshop on Adaptation and Learning in Multi-Agent Systems*.

Mataric, M. M. 1993. Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. *From Animals to Animats II*, ed., Meyer, J. A. and Wilson, S., pp. 432-441, MIT Press, Cambridge.

Meyer, C., Ganascia, J.-G., and Zucker, J.-D. 1997. Learning Strategies in Games by Anticipation. Proc. *International Joint Conference on Artificial Intelligence*, pp. 698-703, Nagoya, Japan.

Mitchell, T. 1997. Machine Learning, McGraw Hill.

Mor, Y., Goldman, C. V., and Rosenschein, J. S. 1996. Learn Your Opponent's Strategy (in Polynomial Time) ! *Adaptation and Learning in Multiagent Systems*, ed., Weiss, G. and Sen, S., pp. 164-176, Springer Verlag, Berlin.

Moukas, A. 1996. Amalthaea : Information discovery and filtering using a multiagent evolving ecosystem. Proc. *PAAM'96*, London, UK.

Moulin, B., and Cloutier, L. 1994. *Collaborative Work based on Multi-Agent Architectures: A Methodological Perspective. Soft Computing: Fuzzy Logic, Neural Networks and Distributed Artificial Intelligence*, ed., Aminzadeh, F. and Jamshidi, M., pp. 261-296, Prentice Hall.

Muggleton, S., and Raedt, L. D. 1994. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19(20): 629-679.

Nagendra Prasad, M. V., and Lesser, V. R. 1997. Learning Problem Solving in Cooperative Multi-agent Systems. Proc. *Working notes of the AAAI Workshop on Multiagent Learning*, pp. 53-58.

Parisi, D., Cecconi, F., Piazzalunga, U., and Denaro, D. 1994. Social Aggregations in Evolving Neural Networks. *Artificial Social Systems (LNAI 830), selected Papers of the MAAMAW'92 Conference*, ed., Castelfranci, C. and Werner, E., Springer-Verlag, Berlin.

Parkes, D., and Ungar, L. 1997. Learning and Adaptation in Multiagent Systems. Proc. *Working notes of the AAAI Workshop on Multiagent Learning*, pp. 47-52.

Plaza, E., Arcos, J., and Martin, F. 1995. Cooperative Case-Based Reasoning. *Adaption and Learning in Multi-Agent Systems*, ed., Weiss, G. and Sen, S., pp. 180-201, Springer-Verlag.

Pont, M.-J., and Moreale, E. 1996. *Towards a Practical Methodology for Agent-Oriented Software Engineering with C++ and Java. 96-33*, Leicester University, Dept of Engineering.

Potter, M. A., DeJong, K., and Grefenstette, J. J. 1995. A Coevolutionary Approach to Learning Sequential Decision Rules. Proceedings *of the sixth International Conference on Genetic Algorithms*, pp. 366-372, University of Pittsburgh.

Quinlan, J. R., and Rivest, R. 1989. Inferring Decision Trees Using the Minimum Description Length Principle. *Information and Computation*, 80: 227-248.

Rachlin, J., Kasif, S., Salzberg, S., and Aha, D. 1994. Towards a Better Understanding for Memory-Based Reasoning Systems, International Conference on Machine Learning , pp. 242-250, New Brunswick, NJ.

Riolo, R. L. 1987a. Bucket Brigade Performance : Default Hierarchy. Proceedings *of the Second International Conference on Genetic Algorithms and their Applications*, pp. 196-201, July 1987, MIT, Cambridge, MA.

Riolo, R. L. 1987b. Bucket Brigade Performance : Long Sequences of Classifiers. Proceedings *of the Second International Conference on Genetic Algorithms and their Applications*, pp. 184-195, July 1987, MIT, Cambridge, MA.

Rumbaugh, J., Blaha, M., Eddy, F., Premerlani, W., and Lorensen, W. 1991. Object Oriented Modeling and Design, Prentice Hall.

Saitta, L., and Neri, F. 1998. Learning in the "Real World". *Machine Learning*, to appear.

Schultz, A. C. 1994. Learning Robot Behaviors using Genetic Algorithms. Proc. *Intelligent Automation and Soft Computing: Trends in Research, Development and Applications, Proceedings Of The First World Automation Congress*, pp. p607-612, Albuquerque, New Mexico.

Schwartz, A. 1993. A Reinforcement Learning Method for Maximizing Undiscounted Reward. Proc. *Tenth International Conference on Machine Learning*, University of Massachussets, Amberst.

Sen, S. 1997. Multiagent Learning. Proc. *Working notes of the AAAI Workshop on Multiagent Learning*, pp. 76, August.

Sian, S. S. 1991. Adaptation based on Cooperative Learning in Multi-Agent Systems. *Decentralized A.I.*, ed., Demazeau, Y. and Muller, J.-P., pp. 257-286, Elsevier Science.

Singh, S. P. 1992. Transfer of Learning by Composing Solutions of Elemental Sequential Tasks. *Machine Learning*, 8(3/4): 323--339.

Smith, R. G., and Davis, R. 1980. *The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem-Solver. IEEE Transactions on Computers*, C29(12).

Stepp, R. E., and Michalski, R. 1986. Conceptual Clustering, Inventing Goal-Oriented Classifications of Structured Objects. *Machine Learning: An artificial intelligence approach*, ed., Michalski, R., Carbonell, J., and Mitchell, T., pp. 471-478, Morgan Kaufmann, Los Altos, CA.

Stone, P. 1996. Multiagent Systems : A Survey from a Machine Learning Perspective. Proc. *IEEE Transactions on Knoledge and Data Engineering (TKDE).*

Stone, P., and Veloso, M. 1996a. Beating a defender in robotic soccer: Memory-based learning of a continuous function. *Advances in Neural Information Processing Systems*, ed., Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E.

Stone, P., and Veloso, M. 1996b. Towards Collaborative and Adversarial Learning: A Case Study in Robotic Soccer. *International Journal of Human-Computer Studies/Knowledge Acquisition*.

Stone, P., and Veloso, M. 1997a. A layered approach to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence (AAI).*

Stone, P., and Veloso, M. 1997b. Using Decision Tree Confidence Factors for Multiagent Control. Proc. *Working notes of the AAAI Workshop on Multiagent Learning*, pp. 71-78.

Stone, P., Veloso, M., and Achim, S. 1996. Collaboration and Learning in Robotic Soccer. Proc. *Micro-Robot World Cup Soccer Tournament (MIROSOT'96)*, November 9-12, Taejon, Korea.

Sycara, K. 1989. *Multiagent Compromise via Negotiation*. *Distributed Artificial Intelligence II*, ed., Gasser, L. and Huhns, M., Morgan Kaufmann.

Takadama, K., and Nakasuka, S. 1996. Emergent Strategy for Adaptive Behaviors with Self-Organizational Individuality. Proc. *The Fisrt Asia-Pacific Conference on Simulated Evolution and Learning*, Taejon, Korea.

Weiss, G. 1996. Distributed Artificial Intelligence Meets Machine Learning: Learning in multi-agent environments. Lecture Notes in Artificial Intelligence, Volume 1221, Springer-Verlag.

Weiss, G., and Sen, S. 1995. Adaption and Learning in Multi-Agent Systems. Lecture Notes in Artificial Intelligence, Volume 1042, Springer-Verlag.

Werner, E., and Demazeau, Y. 1992. *Decentralized A.I. 3.* , North Holland.

Werner, G. M. 1994. Using Second Order Neural Connections for Motivation of Behavioral Choices. Proceedings *of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior*, Cambridge, MA.

Wersterdale, T. H. 1987. Altruism in the Bucket Brigade. Proceedings *of the Second International Conference on Genetic Algorithms and their Applications*, pp. 22-26, July 1987, MIT, Cambridge, MA.

Wooldridge, M. 1992. *The Logical Modelling of Computational Multi-Agent Systems*, PhD thesis, UMIST, Manchester.

Wooldridge, M. J. 1997. *Agent-based software engineering*. *IEEE Proceedings on Software Engineering*, 144(1): 26-37.

Yourdon, E. 1989. *Modern Structured Analysis*, Yourdon Press, Englewood Cliffs, New Jersey.

Zucker, J.-D., Ganascia, J.-G., and Bournaud, I. 1998. Relational Knowledge Discovery in a Chinese Characters Database. *Applied Artificial Intelligence*, Special Issue on KDD in Structural Domains (to appear).