USING THE *CASSIOPEIA* METHOD TO DESIGN A ROBOT SOCCER TEAM

ANNE COLLINOT and ALEXIS DROGOUL
LAFORIA/CNRS—Université Paris VI, Paris, France

We have defined the Cassiopeia *method, whose specificity is to focus the analysis and design of a multiagent system on the notion of organization. This article reports the use of this methodological framework for designing and implementing the organization of a robot soccer team in the context of a research project on collective robotics. We show why we chose this application, and we discuss its interest and inherent difficulties, in order to clearly express the needs for a design methodology dedicated to distributed artificial intelligence.*

The multiagent paradigm is widely used to provide solutions to a variety of organizational problems related to the collective achievement of one or more tasks: computer-supported cooperative work, flexible workshop or network management, distributed process control, or coordination of patrols of drones (Avouris & Gasser, 1992; Werner & Demazeau, 1992; Demazeau & Muller, 1991). All these problems share a common difficulty of design: how to proceed from a global specification of the collective task to the specification of the individual behaviors, which are to be provided to the agents that participate to the task's achievement. A problem of organization has to be solved, most of the time in a dynamic way, so as to obtain the collective achievement of the considered task. In previous works, we have defined the *Cassiopeia* method [Collinot et al., 1995, 1996), the purpose of which is to provide a methodological framework to design multiagent systems (MAS). The underlying principle of *Cassiopeia* is to focus the design activity on the notion of organization. In this paper, we report the use of the *Cassiopeia* method in the context of a research project (Making Intelligent Collective Robotics, or MICROB) on collective robotics.

This article is organized as follows: in the next section we present the MICROB project and the application we are concerned with, that is, the design of a soccer robot team. We discuss the interests and difficulties that characterize this application, which directly set the requirements for a methodology dedicated to distributed artificial intelligence (DAI). Then we give an overview of the methodological approach of *Cassiopeia* and report the use of the *Cassiopeia* method for the design

Revised June 1997.

The MICROB project is conducted by Dominique Duhaut (Robotics Laboratory of Paris) and Alexis Drogoul. We thank Laurent Magnin for providing us with the SIEME simulator and Laurent Ploix for implementing the agents' behaviors described in this article.

Address correspondence to Anne Collinot, LAFORIA/CNRS—Université Paris VI, 4 Place Jussieu, Case 169—75252 Paris Cedex 05. E-mail: {collinot, drogoul}@laforia.ibp.fr

Applied Artificial Intelligence, 12:127–147, 1998

0883-9514/98 $12.00 + .00

and implementation of the soccer robot team. Finally, we discuss our current results and draw conclusions.

MICROB PROJECT AND SOCCER-PLAYING ROBOTS

The aim of the MICROB project (Figure 1) is to investigate collective phenomena of organization in groups of mobile robots. The project relies on the use of a very simple test bed, which enables one to conduct experiments that focus on the software aspects of the agents/robots, especially the ones related to their organizational capabilities.

MICROB Experimental Test Bed

MICROB mobile robots are made of remote-controlled cars, which thus have no sensor and no onboard decision module. Each car receives its commands through a remote-control pilot, which transmits the instructions that have been worked out either by a software agent or a human agent. A camera films the scene, while an image processing system analyzes the position and the speed of all the robots and objects that are on the field. These data are then placed into a simulated environment, within which the software agents are operating to make appropriate decisions that will be sent to the remote-control pilots. Given this test bed, the issue is how to design the software agents, so that the robots they control can exhibit capabilities to organize themselves to collectively achieve a given task.

Application to a Robot Soccer Team

The application to a robot soccer team has been chosen to investigate the design of robots with capabilities to organize themselves to achieve a collective task defined from a common goal. The playground in which the robots are moving is rectangular

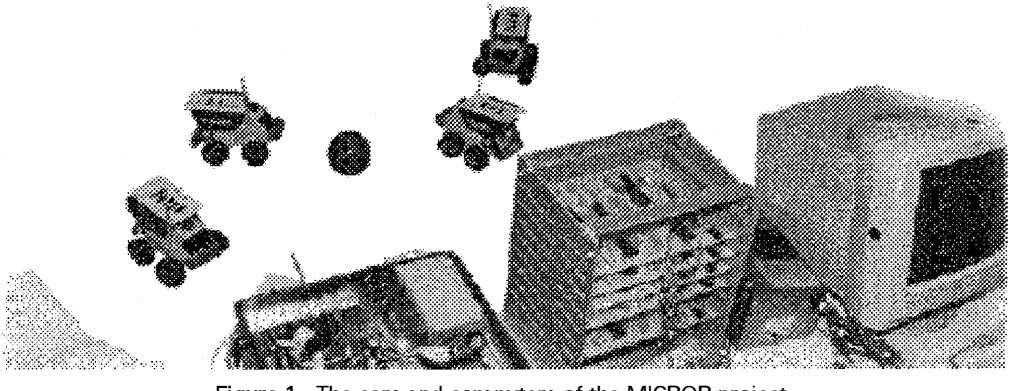

Figure 1. The cars and computers of the MICROB project.

and consists of a goal at each end. There are eight cars on the field, which are distributed into two teams. The cars of one team are remote-controlled by software agents, while the cars of the opposing team are remote-controlled by human agents. The aim of the designer is to enable the team of software agents to score goals while defending its side.

Designing a Robot Soccer Team: Interests and Difficulties

The problem is then to design the software agents so that they can play with "team spirit." Indeed, they should have capabilities to organize themselves in order to win. The main difficulty is to express locally, at the agent level, the behaviors allowing the team to obtain the achievement of the collective task (i.e., the MAS). There are two other difficulties: one is related to the dynamics of the game, which makes it impossible either to define in advance the organization of the robots, or to control the game in a centralized way. The other difficulty derives from the fact that two organizations are facing each other, one being unpredictable (the robot team controlled by human agents).

This test bed is also interesting, since it enables us to cover most of the issues of artificial organizations, with respect to analysis, design, implementation, experimentation, and validation (Kitano et al., 1995). The soccer robot application thus appeared quite appealing to us in the context of developing the *Cassiopeia* framework, by allowing us to structure the design of a large project in a comprehensive manner, and to evaluate the contribution of *Cassiopeia* more seriously than in the case of a toy application.

Requirements for a Methodology Dedicated to DAI

In most cases, a large project, especially a pluri-disciplinary project, requires the use of a methodology. Its main role is to help in identifying the necessary steps that permit us to proceed from the project requirements to their fulfillments (i.e., the project life cycle). Such an approach relies on (1) the use of a homogeneous terminology, which has a meaning at each step of the cycle and which supports correspondence between steps; (2) the use of *operational* conceptual abstractions[1]; and (3) the possibility to backtrack from any step to the previous ones without losing what has been done before.

In the case of DAI, the project requirements consist in having agents achieving a collective task. To fulfill these requirements, one must design the agents with specific attention to their capabilities for organizing themselves. The existing methodologies, especially the object-oriented methodologies (Rumbaugh et al., 1991) that can be considered because of some similarities, such as distribution or locality, provide an interesting basis of analysis: indeed, they enable the distribution of the requirements along both structural and behavioral axes. However, they do not

offer any methodological framework to take the organization issue into account, since it is not considered as an object of analysis in itself. On the other hand, the few DAI works (e.g., Moulin & Cloutier, 1994) that deal with methodological aspects are not very satisfactory, since they only address the organization issue indirectly, through the use of various negotiation or coordination techniques, which in fact, are only particular implementation methods.

The features of the MICROB project stress the need for a methodological framework enabling the integration of both descriptive and operational aspects of the organization. This integration should be made possible, as early as the analysis step, for both implementation and documentation reasons. We thus have decided to use the *Cassiopeia* method, which is presented in the next section.

METHODOLOGICAL APPROACH OF *CASSIOPEIA*

The *Cassiopeia* method (Collinot et al., 1995, 1996) is primarily a way to address a type of problem solving where collective behaviors are to be put into operation through a set of agents. Although *Cassiopeia* does not yet offer all the components that one could expect to find in a complete method (as in object-oriented methods like the Object Modeling Technology (OMT) (Rumbaugh et al., 1991), it provides a methodological framework that permits us to better understand and plan the design of a computational organization. According to *Cassiopeia,* a MAS should be designed in terms of agents provided with three levels of roles: the domain-dependent roles, the relational roles, and the organizational roles. This enables the identification of various types of choices made at precise points in the design process. The modification of a MAS is then systematically related to the revision of these choices.

The underlying principle of *Cassiopeia* is that the problem of organizing individuals to achieve collective problem solving must be addressed as such by the designer and/or the agents. Such an organizational issue arises, since *functional dependencies* are inherent to the collective achievement of the considered task: the activation of an individual problem-solving behavior affects and is affected by the behaviors activated by other agents. This whole set of dependencies determines the *coupling* of the organizational problem underlying the collective task achievement. We distinguish two types of coupling. (1) When the individual behaviors are not competing, the coupling is static, that is, the organization of the agents remains unchanged and the designer can define it in advance (e.g., the multi-expert systems in which there is exactly one expert per domain of expertise). (2) When some sort of competition is to be managed (individual behaviors exist that are equivalent for a given situation, or a same behavior can be operated by different agents), the coupling is said to be dynamic, and the organization cannot be defined in advance, since it depends on the context. Therefore the designer can only consider the

organizational structures between agents that will be instantiated within the problem-solving context. This is the case in our application, where the soccer robots have to organize themselves dynamically.

The *Cassiopeia* method proceeds from the collective task to the design of the MAS along three steps that reconcile both local and global views by integrating bottom-up and top-down approaches[2] (Figure 2): (1) definition of the domain-dependent roles that are required by the task; (2) structural description of the organization resulting from the dependencies between the roles and definition of the relational roles; (3) description of the organizational dynamics and definition of the organizational roles. The black arrows in Figure 2 represent the bottom-up (transition from local to global viewpoints) and top-down (transition from global to local viewpoints) paths that the designer should follow to proceed along the *Cassiopeia* steps.

Step 1: Definition of Domain-Dependent Roles

The identification of the elementary behaviors that should be put into operation by the agents to achieve the considered task does not come under *Cassiopeia:* most of the time, it results from a functional (e.g., Yourdon, 1989) or object-oriented (e.g., Rumbaugh et al., 1991) analysis step. Given these behaviors, the first step of *Cassiopeia* consists in identifying the required level of abstraction so that those behaviors make sense with respect to the *collective* achievement of the task: the

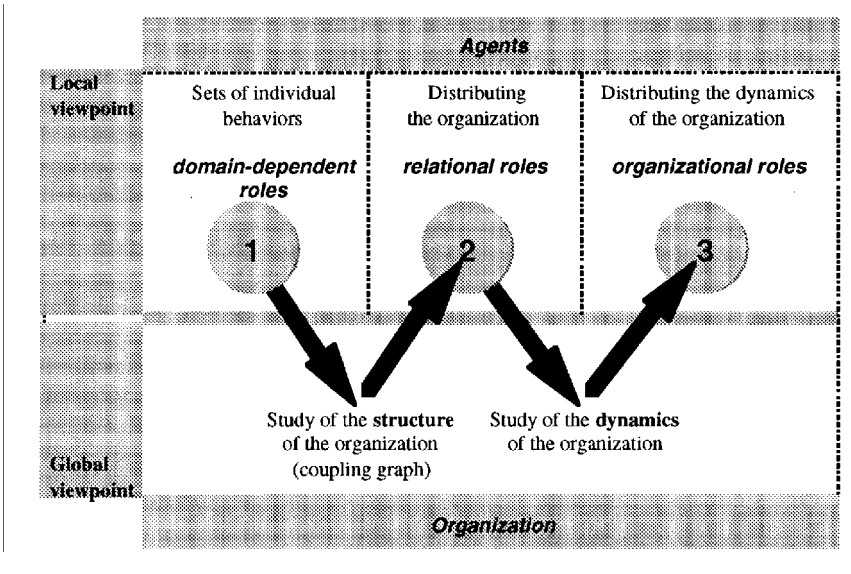

Figure 2. Steps of the *Cassiopeia* method.

designer specifies the sets of behaviors that achieve proper functionalities and thus determines the domain-dependent roles that the agents can play.

In most cases, the definition of the appropriate roles needs to proceed 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. Indeed, the notion of role is an ambivalent concept, which enables us to represent 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 process of role definition requires special care, in that the following design operations will largely rely on it. However, it is possible to revise at any time the choices that have been made, by adding new roles or removing or modifying the existing roles.

The designer defines the agents by the set of roles they can play. The domain-dependent roles that have been specified are not necessarily provided to all the agents. Some constraints related to their possible behaviors, along with some design choices, can lead to specialize some agents, that is, to provide them with only a subset of the available roles. The extreme case is to have only one role per agent.

Step 2: Definition of Relational Roles

Definition of Coupling Graph

The second step consists in analyzing the organization structure based on the dependencies between the considered domain-dependent roles. Such dependencies can be functional, when they derive from the dependencies that exist between the behaviors implemented by these roles, or they can be 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, for example, Figure 4). The identification of these dependencies leads to define the *coupling graph* underlying the considered collective task. 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 paths and the elementary circuits of this coupling graph define the potentialities of grouping the different domain-dependent roles together and therefore provide a global representation of the structure of the organizations to which the agents can belong when playing these roles.

Notion of Influence

The dependencies between the domain-dependent roles are naturally translated into dependencies between the agents that can play these roles. In order for the agents to determine their roles depending on those played by the other agents, one should provide them with some means for identifying and handling these dependencies.

For this purpose, *Cassiopeia* resorts to the 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 *relational roles.* An influencing agent produces *signs of influence* that correspond to the domain-dependent role it is playing; an influenced agent must be able to interpret these signs to play the domain-dependent role that corresponds to the exerted influence. The definition of these two relational roles therefore enables us to distribute the structure of the organization (as defined by the coupling graph of the considered task) among the agents, as a transmission of influence signs. At this step, influence signs that come from sources other than the agents, as for example, those produced by the environment, can also be taken into account.

It is to be noted that the choice of the techniques that respectively permit us to produce and interpret an influence sign does not come under the scope of *Cassiopeia.* However, it is worth pointing out that the necessary communication or interaction mechanisms should be chosen at the time of the specification of the influence signs.

When setting the relational roles, the designer must also specify the (relational) behaviors that permit us to control the domain-dependent roles (for instance, how to choose among several influences). In this view, it is possible that the distinction between only two types of relational roles ("influencing" or "being influenced") appears somewhat restrictive. When this is the case, it is possible to adopt a qualitatively richer typology, such as the *social dependencies* typology introduced by Castelfranchi (1992) and further developed by Sichman and Demazeau (1995).

Step 3: Definition of Organizational Roles

The two previous steps allow us to define agents able to (1) play the domain-dependent roles required by the task achievement and (2) exert a control upon those roles (for instance, inhibit, activate) depending on the roles played by the other agents. At this time, all the potentialities for groups of agents to take place are present. The third 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 groups: the roles of *initiator* and *participant.*

The behaviors that are associated to the relational roles may allow the formation of several groups that are redundant with respect to their end results. When such a redundancy is useless or costly, it is necessary to determine the criteria that affect the choices to form one group rather than another one. 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. Indeed, an initiator agent plays a domain-dependent role that makes it belong to all the groups it can potentially form with the agents playing roles that are depending on its own. 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 must thus (1) identify the agents likely to play a role of initiator (according to the grouping potentialities that have been identified in the coupling graph) and (2) determine, for each of them, the selection methods allowing 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 participant consists in activating *commitment behaviors.*

Group Formation Behaviors

Group formation behaviors are specified to enable an initiator agent to dynamically organize a collaboration with other agents. Basically, these behaviors are aimed at selecting the most appropriate agents to collaborate with.[3] The corresponding selection methods can be implemented by a variety of techniques (Decker, 1987) based on task announcement such as all the techniques deriving from the Contract Net (Smith & Davis, 1980), the notion of consensus and negotiation among agents belonging to concurrent groups (Sycara, 1989; Rosenschein & Genesereth, 1985); priorities allowing us to order the potential groups (Erdman & Lozano-Perez, 1986); or the use of a supervisor or a hierarchy (Le Pape, 1990).

Commitment Behaviors

Next, the designer specifies the *commitment signs,* which enable the initiator agent to indicate to the selected agents that a group is formed with them. These agents are then going to play a role of participant in this group. 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 keep committed to the group—as long as it is maintained.

Dissolution Behaviors

Finally, the designer must consider the dissolution of a group. Indeed, the choices resulting from the group formation behaviors may need to be revised. 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 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.

Figure 3 sums up the three levels of roles that are defined by *Cassiopeia,* along with the different types of associated behaviors and signs.

Necessity of Organizational Roles

It is important to note that the third step is not always necessary, especially in the case of self-organized systems or systems that are organized once for all time (the two extremes on the axis of possible organizations).

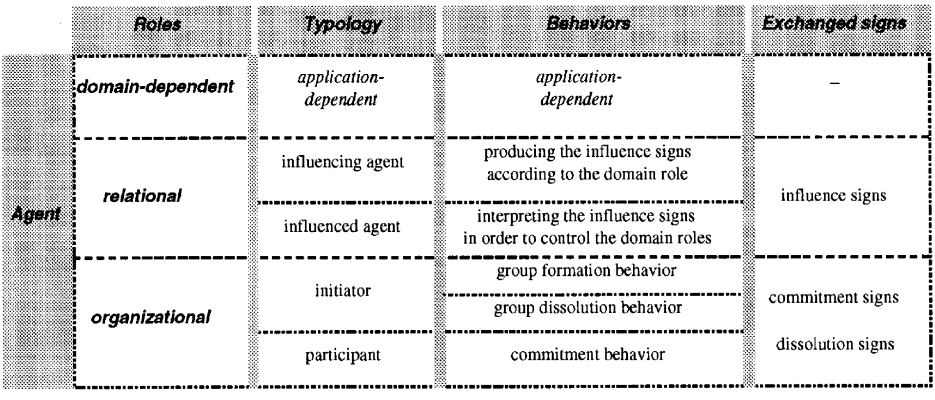

Figure 3. *Cassiopeia* summing up.

In the first case, one expects the organizational roles to appear as "emergent constraints" upon the relational roles. This is the case, for example, of the simulated ants of the MANTA system (Drogoul et al., 1995). They do not have any organizational roles, but the evolution of some parameters defined at the relational level enables them to specialize in particular influences as the simulation is going on. In this way, the ants become specialized in particular domain-dependent roles and therefore "self-attribute" a role in an organization that only exists in the observer's mind. Agents with such behaviors (able to play domain-dependent and relational roles but not organizational ones) are often called *reactive agents* in DAI terminology.

Conversely, in an organization where the organizational roles are distributed in advance, as for example, in a strict hierarchical structure, the agents do not have to activate explicit behaviors when playing their organizational roles: the organization is static, since the influence relationships are defined in advance and do not depend on the problem-solving context.

USING *CASSIOPEIA* TO DESIGN A ROBOT SOCCER TEAM

In this section, we report how we followed the steps of the *Cassiopeia* method to design the multiagent system used to pilot the MICROB robot soccer team. The point of this section is more to follow the method step by step rather than to give the details of the implementation of the end-user application. This is why we have decided to use insofar as possible "standard" DAI techniques, especially for defining the behaviors associated to the relational and organizational roles.

The interest of this section does not then reside in the algorithmic solutions (although the application is performing quite well) but in the way these solutions are introduced by (and integrated in) the design process. As we shall see in the section

below, Preliminary Results and Work in Progress, very different options can be considered without questioning the design process.

Step 1: Domain-Dependent Roles

The first step has consisted in defining which roles should be played by the different robots to (1) ensure that essential functionalities, such as to defend or shoot the ball, are provided; and (2) see to it that the robots really practice a collective game, based on the correct circulation and sharing of the ball. This led to define four domain-dependent roles:

- the role of shooter, which consists in shooting the ball (in the direction of the opposing goal or a teammate);
- the role of placer, which consists in placing oneself on the playground to receive a pass;
- the role of blocker, which consists in blocking an opponent's way;
- the role of defender, which consists in preventing its own side from any opposing intrusion.

Playing a role leads a robot to adopt a specific behavior, which need not be described in detail in this article. However, it is important to stress that such behavior is not *elementary* with respect to robotics concerns (one would expect elementary robot behavior to be "turn left," "accelerate," etc.). It is implemented as a more or less sophisticated combination of these elementary behaviors. One should also note that these roles are defined with respect to a *collective* game, although no procedure for playing together is provided. In this way, for example, playing the role of placer makes sense for an agent only if other agents can make a pass to him.

In the current state, each robot can play the four roles. We decided not to define specialists (for instance, the goal keeper, which would only play the role of defender), so as to increase the possibilities of dynamic organization of the group. Each of the behaviors is exclusive, which means that, at any time, an agent has one active role and three idle roles. Obviously, the active role is determined both by the context and the roles played by its teammates. As we shall see in the next sections, such dynamic control is handled by the relational and organizational roles.

Step 2: Relational Roles of Robots

The four soccer roles are strongly dependent on one another. For instance, the shoot of a robot, that is, the behavior it activates when it plays the role of shooter, depends on how the other robots are placed in the playground: one should not make a pass to a misplaced robot; neither shoot anywhere. More generally, it is obvious that the very structure of the play relies on such dependencies. In the *Cassiopeia*

method, this is translated in the coupling graph shown in Figure 4 (on the diagram, an arrow from role A to role B means that role B potentially depends on role A).

We now describe the various types of dependencies between the roles, that is, the dependencies that are neither redundant, inconsistent, nor useless with respect to the interactions between the agents playing these roles:

d1 Defending depends on the role of defender of the other agents (position on the playground, etc.);
d2 Shooting can enable oneself to shoot again (this is the pass to oneself) or another robot to shoot (this is the standard pass);
d3 Shooting the ball depends on the position of the placers;
d4 Defending or being well placed can enable a robot to intercept the ball of the opponent (and thus to shoot in a further step);
d5 Blocking an opponent can enable another robot to shoot.

This choice of dependencies is clearly discretionary, but it is explicit and we know where it has been done in the design cycle. The coupling graph and the definition of the dependencies are used to specify the influence signs that are produced by the agents to instantiate these dependencies in the context of a collective

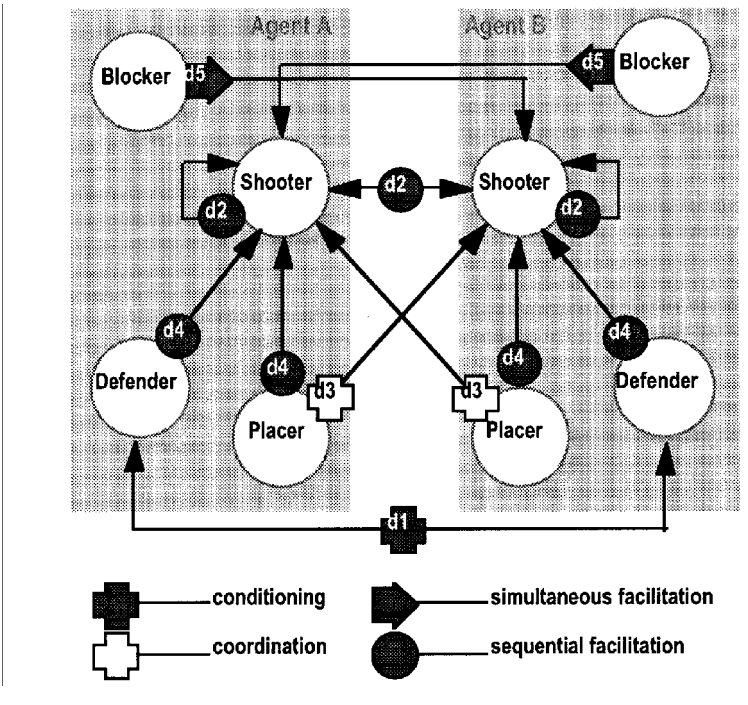

Figure 4. Coupling graph of the football game.

game. Once again, the implementation choices do not interfere with the design process. In particular, in the current implementation the communication protocol is implemented through a simple object-oriented synchronous[4] message passing. In other domains, it could be implemented through asynchronous message passing, broadcast of information, writing on a blackboard, etc. It is not the purpose of *Cassiopeia* to detail this aspect, which is left to the designer.

In our MICROB project, the *relational roles* (being influent toward other agents, or being influenced by other agents) are thus entirely implemented as the *sending of influence messages* that are specific to the domain-dependent role played by the influencing agent, and the *execution of the associated methods* by the receiver (the influenced agent). In this way, the influence relationships between the agents are handled by the production and the interpretation of four types of influence messages (which correspond to the dependencies of the graph shown on Figure 4).

- The *help* message (corresponding to **d5**): A blocker can provide help to a shooter. Any shooter is able to send a *help* message, which can be interpreted by any agent to return a value representing the receiver's capacity to block, based on its local perception of the game.
- The *shoot* message (corresponding to **d2**): A shooter may make a pass to itself or another shooter. It sends a *shoot* message, and the receivers of the message return their capacity to shoot. It then adjusts the direction of its shoot according to the received information.
- The *place* message (corresponding to **d2, d3, d4, d5**): A shooter must take the other robots' positions into account before shooting the ball. The *place* message is sent by the shooter, and the receivers of the message return their capacity to shoot and place themselves in a relevant fashion.
- The *position* message (corresponding to **d1, d4**): The defenders influence each other robot by their position on the playground. The *position* message is sent by a defender, compelling each receiver to indicate its (absolute or relative) position, which the sender uses to adjust the direction of its move.

Once the system of influence relationships is set up, it remains to specify the relational behaviors that enable us to handle the nonsocial influence signs, that is, (1) the absence of influence, which constitutes a sign in itself, and (2) the environmental contexts that the agents consider as influence signs. In MICROB the absence of influence leads any agent to adopt the role of defender, and the presence of the ball to adopt the role of shooter. The next section describes how the groups of agents will dynamically form based upon these influence relationships.

Step 3: Organizational Roles of the Robots

The behaviors allowing the agents to organize themselves according to their respective roles are based on a standard contract net technique (Smith & Davis,

1980). The organizational roles of initiator and participant are thus translated into a *manager* and a *bidder* behavior. When analyzing the coupling graph (Figure 4), it appears that the role of shooter depends in different ways on all the other roles. A shooter is therefore likely to belong to several groups, and we decided to provide him with the role of initiator.[5]

- The formation behaviors are then only defined for the shooter agents: when an agent becomes a shooter, it evaluates the capabilities of its teammates to place themselves and block opponents. It enters into a placing contract and a shooting contract with those that return the best bids. It also evaluates the capabilities of the other robots to shoot. When a robot is better placed than itself, it enters into a shooting contract with this robot.
- The associated commitment signs are simply the names of the domain-dependent roles the participant agents have to play when a contract is concluded. The initiator agent sends these signs by way of messages.
- The commitment behaviors of a participant consist in inhibiting its relational roles so as to only pay attention to the influence signs related to the domain role it has to play. In this way, it closes itself to any other influence as long as it is committed to the group.
- The dissolution behaviors are exclusively defined for the shooter. A shooter activates this type of behavior when it enters into a shooting contract with another robot or manages to score a goal. It breaks off all its current contracts and sends a specific dissolution sign, allowing the other robots to "release" their relational roles in order to be open again to any type of influence.

Figure 5 sums up the different steps of the *Cassiopeia* method as we applied it to the design of the soccer robots.

The setting up of the organizational roles allows us to make the desired changes of domain role appear. Figure 6 shows how two robots change roles so that the one with the best capability to shoot the ball is the shooter. In a first step, robot A plays the shooter role. It enters into a placing contract with robot B. After the shoot by robot A, robot B has the best capacity to shoot. Robot A comes to a shooting contract with B. The placer (B) becomes a shooter, and the shooter (A) becomes a defender. The new shooter (B) immediately enters into a placing contract with the defender (A), which then becomes a placer. In this way, a dynamic reorganization occurs around the new shooter (B), which then shoots the ball and scores a goal.

PRELIMINARY RESULTS AND WORK IN PROGRESS

In this article, we have reported the use of a DAI-oriented methodological framework to design the organization of a robot soccer team. This design work has resulted in the implementation of a system of software agents that operate within a

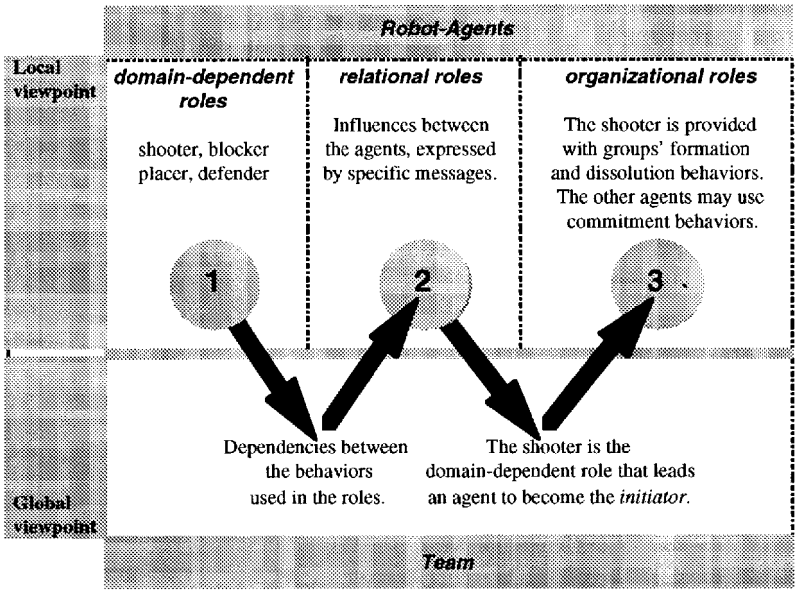

Figure 5. The *Cassiopeia* method applied to the soccer robot design.

virtual environment. This environment is computed based on the playground in which the robot-cars that the agents remotely control are moving. Since it is long and costly to conduct physical experiments, we developed a first prototype using the SIEME simulation framework (developed in our laboratory). This allowed us to carry out a preliminary evaluation of the roles and behaviors that have been designed

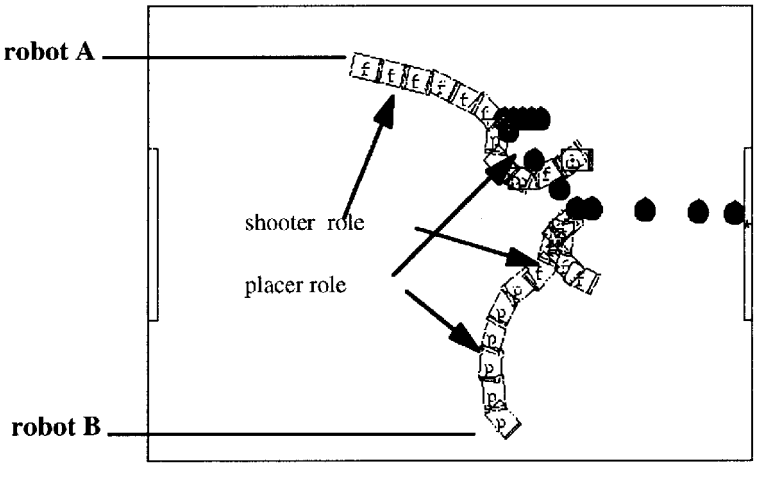

Figure 6. Dynamics of a simple two-player game.

and implemented according to *Cassiopeia*. The experimental results have been very satisfying in terms of both the agent's individual capability to play efficient roles on the playground, and the collective capability to carry on offensive or defensive joint actions. Obviously, it is difficult to interpret these results from a statistical point of view, since our team has opposed either teams remote-controlled by human agents, or other similar robot teams. We intend to make the first life-size tests at the time of the world soccer robotcup (Kitano, 1995), which will be held every year as of November 1996 and will gather a number of robot teams from all over the world.

However, we are already able to evaluate the benefits we have gained from the use of our design method. Indeed, such a method is generally appreciated in terms of two qualitative criteria: the reusability provided by the method genericity and the robustness, that is, the method capacity to tolerate the revision of decisions related to analysis, design, or implementation. While it is still early to evaluate the reusability of our method, we can illustrate the robustness of *Cassiopeia*. In the next sections, we show how we have easily adapted our methodological approach to different choices of agent architectures.

Removing Communication Between Agents

Our main purpose was to gradually reduce the hardware and software devices required to implement the robot behaviors described in the previous sections. Indeed, it is important to save as much time as possible, since the environment is highly dynamic.

The first change consists in removing, purely and simply, all the possibilities to communicate between the agents. In this way, the transmission of influence, commitment, or dissolution signs, which was implemented by a mechanism of message sending, becomes irrelevant. However, the position of each robot is known by every other robot, and each robot shares the same abilities to play the roles (defined at the three levels advocated by *Cassiopeia*). Based on these facts, we have provided each agent with a complete representation of the game, including its teammates. In this manner, none of the choices made during the application design is required to be revised; only the implementation has been modified: agents do not interact with their teammates, but with their own representations of their teammates, to which they ascribe the roles that they would play if they were in their context. All the necessary communications defined at the relational and organizational levels now take place in each of the simulated worlds attached to the agents rather than between the robots themselves. Obviously, from the moment the world model attached to each agent is accurate, the experimental results remain the same. This work allowed us to investigate the collective game of the combined team (comprising robots that are remote-controlled by software and human agents), where the behavior of the human agents is interpreted by the software agents.

Implementation of Reactive Behaviors

When analyzing the implementation of the system described above, we find that (1) the only piece of information available to the agents is the position of the other agents, (2) the signs, especially those related to the influence relationships, are no longer transmitted with a communication mechanism, but individually computed by each of the receivers (that is, the agents playing the relational role of being influenced). This led us to design agents that are simpler than the previous ones, with the aim to determine the lowest level of simplicity that can be used for individual processing of information while maintaining good collective properties.

We thus designed agents that are able to (1) activate the domain roles described in the section above, Using *Cassiopeia* to Design a Robot Soccer Team, and (2) self-attribute roles without going through an initiator agent, based on the use of an evaluation function of their local context (positions of their teammates, positions of the opponents, position of the ball, etc.). In this view, the relational roles and their associated behaviors are entirely represented by this evaluation function. The need for explicit references to the organizational roles and, in particular, to group formation behaviors is no longer necessary. According to the definition given in the section above, Necessity of Organizational Roles, the newly designed agents have reactive behaviors, and consequently, it is expected that a team of such agents self-organizes; that is, the simple activation of their relational roles is sufficient to generate a collective configuration, equal to or even more efficient than the configurations obtained with the earlier implementations.

As is shown in Figure 7, the first step of the method remains unchanged: the analysis in terms of roles and dependencies is still valid; the relational roles still rely on the considered dependencies, but the way they are implemented is modified;[6] finally, the organizational roles are not explicit anymore. The important point is that the work achieved to define the domain-dependent roles, which required an important effort of analysis, is reusable as it is, although the multiagent system relies on an organizational paradigm that is the opposite extreme of the one we presented in the section Using *Cassiopeia* to Design a Robot Soccer Team.

Conceptually, such a methodological tool allows us to (1) relate organizational mechanisms or techniques that have been clearly differentiated up to now [at least in the literature (e.g., Chaib-draa, 1987)], (2) provide a common conceptual framework, and (3) consider their use only according to their appropriateness to the individual capabilities of the agents, rather than to decisions that are made a priori. The use of a method that recommends making technological decisions at clearly defined levels allows us to relativize the "mysterious" part of emergent organizations in reactive systems: the self-organization of the group is nothing but a more or less correct self-allocation of the roles, in which the observer can recognize (or not) organizations that are appropriate for solving the problem. The teams of reactive agents are currently under evaluation using our simulator. The first games are

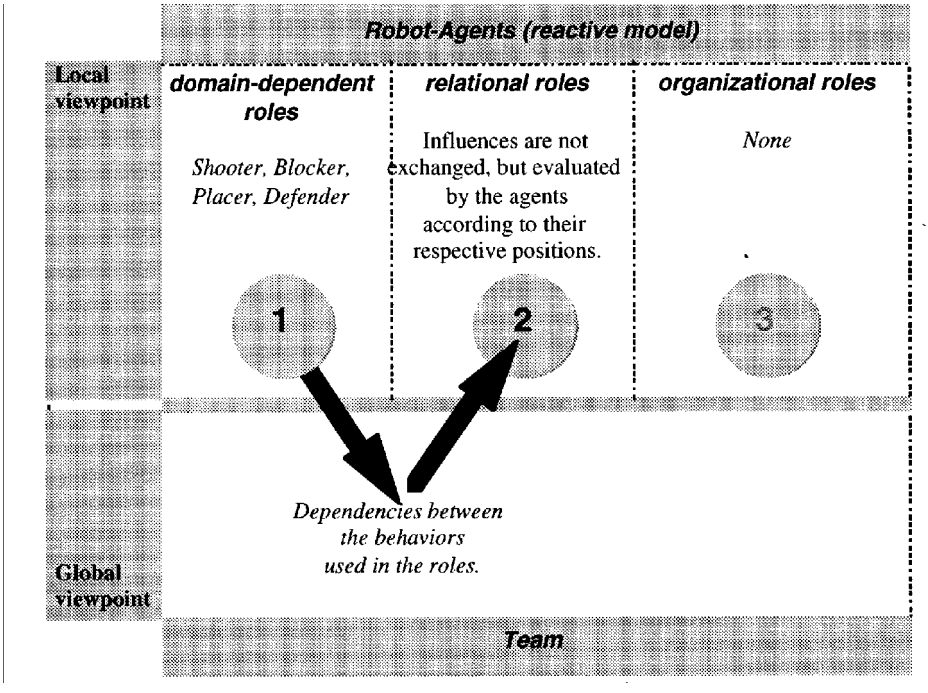

Figure 7. Design of a reactive robot team.

particularly encouraging, although none of these teams has won against the previous defined teams.

Learn to Play with "Team Spirit"

The collective game of a purely reactive robot team is, in fact, relatively stereotyped: for a given situation it is always the same, even when it turns out to be inappropriate. Conversely, it happens that collective responses to unpredicted situations are qualitatively surprising, especially for facing teams that are remote-controlled by human agents. In both cases, the issue is that the robots do not take advantage of their experience of the collective game. They have no available mean to adapt their relational behaviors (that is, the way they handle the influences) with respect to previously encountered situations.

We thus naturally come to the conclusion that providing the agents with some learning mechanisms could allow appropriate organizations to emerge. Such mechanisms should enable the agents not to learn the behaviors associated to the domain-dependent roles that have been identified for soccer (which is not our purpose), but to learn the individual capability to coordinate its domain role with the roles played by the others. Figure 8 gives the design process we have followed to

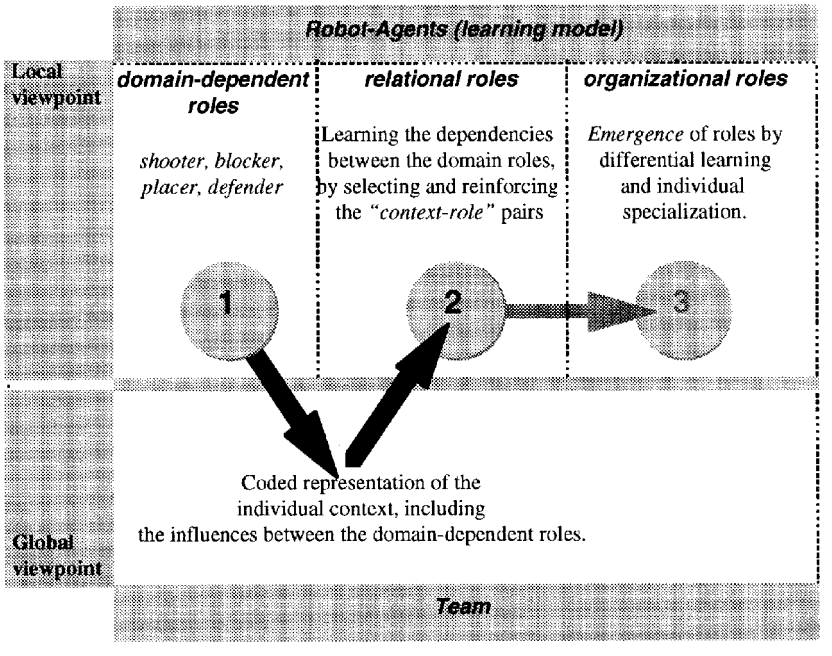

Figure 8. Design of a learning robot team.

build such a system: it is based on the earlier definition of the four domain-dependent roles. The relational roles now consist in selecting the appropriate[7] "perceived context/role to play" pairs based on a coding of the local context of the agents, that is, in rebuilding the dependencies between the domain roles. Although the organizational roles are not specified, we indicate that the goal is to have them "emerge" as a consequence of the specialization of some agents in particular relational roles.

This work is not yet in an evaluation phase, and it is obvious that the experiment protocols should be rigorously defined to evaluate its significance. However, the use of *Cassiopeia* allowed us to precisely determine where the learning mechanisms should be placed among the various types of agents behaviors, that is, to answer the question, What should the agents learn to play with "team spirit?"

The three systems presented in this section demonstrate the benefits gained from applying an operational and methodological approach to the organization of agents. In this way, we have been able to cover a range of coordination or task allocation techniques, from reactive coordination to contract net mechanisms. Moreover, this has been achieved while keeping the same analysis and retaining a large part of the design process by-product. The next step will consist in comparing the various teams we have designed with regard to the quality and efficiency of their game.

CONCLUSION

The current state of the MICROB project allows us to distinguish three major contributions of the *Cassiopeia* method:

- Allowing to manipulate, in an homogeneous way, conceptual abstractions, such as roles, organizations, or influence relationships, from the analysis step to the implementation step;
- Facilitating the interfacing of the robotician's and computer scientist's languages by providing a conceptual level that is neither too sketchy nor too technical; this is certainly due to the use of an operational multiagent-oriented terminology where the justification of each term is grounded in the role it plays within the method (and not only in subjective sociological or biological metaphors);
- Clearly distinguishing, at design time, between the roles and behaviors that are domain-dependent and the ones that are relevant to the agent's organization; this is made possible by the three levels of design of *Cassiopeia:* the domain-dependent roles, the relational roles, and the organizational roles. The use of these three levels appeared very helpful.

However, the use of the method has also pointed out the issues that need to be further developed so that it can provide a complete DAI methodological framework, both for research and development.

- We need to refine the current terminology, especially when the terms are used by different DAI researchers with various meanings.
- It is necessary to integrate *Cassiopeia* as a design method to existing analysis methods (in particular, object-oriented methods), which allow us to make the necessary choices related to a domain-dependent structural and behavioral distributions. This should permit us to clarify the status of *agent* as opposed to the status of *object.*
- It is necessary to determine more precisely how to integrate the existing techniques related to communication protocols, negotiation or coordination mechanisms, agent architectural components, etc. We should take advantage that surveys of these techniques are already available (Bouron & Collinot, 1992; Chaib-draa, 1992; Decker, 1987).
- It is also necessary to consider the automation of *Cassiopeia* so as to (1) provide the design with guidelines to proceed from one step to another, (2) propose for each step a set of choices related to the two points above, (3) generate a documentation of the project, and (4) facilitate the reuse of former projects.

All these improvements should enable *Cassiopeia* to evolve toward a complete *agent-oriented design* methodology to develop ambitious multiagent projects, and reuse more efficiently the existing tools and techniques.

NOTES

1. In the case of MASs, it means to give an operational definition (that can be used directly) to descriptive terms that generally stem from sociological or biological metaphors.
2. As an example, what we intuitively call "team spirit" above requires us to provide the agents with individual behaviors such that (1) all the necessary roles for playing soccer are assumed within the team (bottom-up approach) and (2) the constraints resulting from the necessity to dynamically organize themselves during the game are taken into account (top-down approach).
3. However, whether or not the agents should have capabilities to make these choices does not come under the scope of *Cassiopeia.* When the designer decides to provide them with such capabilities, the techniques he will choose depend on the design choices already made for the final application (e.g., choices related to communication and perception). *Cassiopeia* allows us to identify the organizational redundancies and to analyze their effects on the problem-solving quality.
4. An influence sign is a message sent to an agent. Its interpretation by the receiver consists in executing the corresponding method, which returns a value to the sender.
5. Other solutions have been implemented with two types of initiator: the shooter that organizes an attacking group and the defender, a defending group. They are not presented here for reasons of length and clarity.
6. This implementation consists in computing attractive or repulsive potential fields "named after" each of the domain-dependent roles, based on the positions of all the players.
7. To implement these behaviors, we decided to use a system based on classifiers that are selected by genetic algorithms (Holland & Reitmann, 1978).

REFERENCES

Avouris, N., and L. Gasser (eds.). 1992. *Distributed AI: Theory and praxis.* Norwell, Mass.: Kluwer Academic.

Bouron, T., and A. Collinot. 1992. SAM: A model to design computational social agents. In *Proceedings of the Tenth European Conference on Artificial Intelligence,* pp. 239–243.

Castelfranchi, C., M. Miceli, and A. Cesta. 1992. Dependence relations among autonomous agents. In *Decentralized A.I. 3,* eds. E. Werner and Y. Demazeau. New York: North-Holland.

Chaib-draa, B. 1992. Distributed intelligence: An annotated bibliography. *SIGART Bulletin* 3(3).

Collinot, A., P. Carle, and K. Zeghal. 1995. *Cassiopeia:* A method for designing computational organizations. Presented at the First International Workshop on Decentralized Intelligent Multi-Agent Systems, Poland.

Collinot, A., A. Drogoul, and P. Benhamou. 1996. Agent oriented design of a soccer robot team. Presented at the Second International Conference on Multiagent Systems, Japan.

Decker, K. S. 1987. Distributed problem-solving techniques: A survey. *IEEE Transactions on Systems, Man and Cybernetics* 17(5).

Demazeau, Y., and J.-P. Muller (eds.). 1991. *Decentralized A.I. 2.* New York: North-Holland.

Drogoul, A., B. Corbara, and S. Lalande. 1995. MANTA: New experimental results on the emergence of (artificial) ant societies. In *Artificial Societies,* eds. N. Gilbert and R. Conte. UCL Press.

Erdmann, M., and T. Lozano-Pérez. 1986. On multiple moving objects. Presented at the IEEE International Conference on Robotics and Automation.

Holland, J. H., and J. S. Reitmann. 1978. Cognitive systems based on adaptive algorithms. In *Pattern directed inference systems,* eds. D. A. Watermann and F. Hayes-Roth. San Diego, Calif.: Academic.

Kitano, H., A. Minoru, Y. Kuniyoshi, I. Noda, and E. Osawa. 1995. RoboCup: The robot world cup initiative. Presented at the Workshop on Entertainment and AI/Alife, IJCAI.

Le Pape, C. 1990. A combination of centralized and distributed methods for multiagent planning and scheduling. Presented at the IEEE International Conference on Robotics and Automation, Cincinnati.

Moulin, B., and L. Cloutier. 1994. Collaborative work based on multiagent architectures: A methodological perspective. In *Soft computing: Fuzzy logic, neural networks and distributed artificial intelligence,* eds. F. Aminzadeh and M. Jamshidi, pp. 261–296. Englewood Cliffs, N.J.: Prentice-Hall.

Rosenschein, J., and M. Genesereth. 1985. Deals among rational agents. Presented at the 9th International Joint Conference on Artificial Intelligence.

Rumbaugh, J., M. Blaha, F. Eddy, W. Premerlani, and W. Lorensen. 1991. *Object oriented modeling and design.* Englewood Cliffs, N.J.: Prentice-Hall.

Sichman, J., and Y. Demazeau. 1995. Exploiting social reasoning to deal with agency level inconsistency. Presented at the First International Conference on Multiagent Systems, San Francisco.

Smith, R. G., and R. Davis. 1980. The contract net protocol: High-level communication and control in a distributed problem-solver. *IEEE Transactions on Computers,* C29(12).

Sycara, K. 1989. Multiagent compromise via negotiation. In *Distributed artificial intelligence II,* eds. L. G'sser and M. Huhns. San Mateo, Calif.: Morgan Kaufmann.

Werner, E., and Y. Demazeau (eds.). 1992. *Decentralized A.I. 3.* New York: North-Holland.

Yourdon, E. 1989. *Modern structured analysis.* Englewood Cliffs, N.J.: Yourdon Press.