Negotiation and Conflict Resolution in a Multi-Agent System for an Assembly Robotics Cell

Eugénio Oliveira
Ana Paula Rocha
Faculdade de Engenharia Universidade do Porto
Rua dos Bragas, 4099 PORTO codex
PORTUGAL
email: eco@fe.up.pt; Fax: +351 2 319 280

ABSTRACT

A Multi-Agent System Architecture is here both described and justified for the sake of its application to an Assembly Robotics Testbed. A Blackboard-based Agent using a GBB/DBB facility for intra (and not inter) Agent communication and including knowledge about all the agents community, is presented as well as its main functionality. These Agents are well suited for pursuing their own goals through their local activity control as well as the system overall goals by means of appropriate cooperative behavior. Coordination of different Agents dynamically playing the roles either of Organizers or Respondents may lead to the use of both Negotiation and Client/Server protocols for Cooperation. Agents coordination activity regarding conflict resolution is also presented and illustrated through a documented example in the Robotics Testbed. A real Testbed, whose Agents are here briefly presented, working with real-time constraints has been already implemented and tested at our Laboratory.

1 - INTRODUCTION

Intelligent Robotics is one of those domains where hard problems have to be tackled down by means of both symbolic knowledge and complex computations, hence making the Distributed Artificial Intelligence approach attractive, if not mandatory.

Multi-Agent Systems (MAS) architecture - upon which ARCHON[1] philosophy has been based - may be seen as providing Open System characteristics, mainly in which flexibility as well as reliability concerns, therefore enabling continuity of operation while the environment is dynamically changing.

Modeling real world, sophisticated application domains through Multi-Agent Systems, presents some other important advantages like:

* providing a more natural representation of the real world domain expertise, which is often itself distributed;
* breaking down both the problem and the problem-solving strategy into smaller units which makes the search for the solution process more simple and understandable.

A society of Agents avoiding centralized control has also to solve the problem of the relationships between, on one hand, each Agent's behavior and goals and, on the other hand, those of the global System or Multi-Agent community. In order to make the all activity to converge to the global system satisfaction, despite of the Agents degree of autonomy, special policies are to be designed for:

* Agents actions **Coordination**
* Agents **Communication**
* Agents **Cooperation**

Coordination can be dealt with either through (more or less distributed or centralized) planning algorithms, or through the designing of flexible policies and protocols to dynamically direct all the needed agents interactions.

Communication is a technical problem depending on the available operating mechanisms for agents processes interaction. Following the Archon approach our Agents are provided with a Session Layer which makes Unix message passing useful for different kinds of possible Agents interaction policies. Nevertheless, in our opinion, MAS technology and formalisms still lack of a complete framework for information, as well as knowledge and intentions, communication between different, and heterogeneous, entities (Agents, objects, processes) which take into account correctness, efficiency as well as reliability.

Cooperation is described by a set of actions and intentions which deals with the social and cognitive aspects of benevolent Agent interactions. Nevertheless, even discarding non-benevolent Agents, one has to take into consideration the all set of possibilities agents may encounter during their joint activity, when looking for the satisfaction of the system global goal. Dynamic dependency, negotiation, resolution of conflict situations at different levels (on task assignment, on resources utilization and on different perspectives upon partial solutions) have to be modeled and represented in order to give the kind of social behavior they need in order to participate on complex team problem solving.

---

[1] ESPRIT Project "Architecture for Cooperative on-line Systems"

Work we have done in the framework of the Archon project, have contributed to design methodologies as well as strategies responsible for Multi-Agent Systems activity in cooperative domains. This paper briefly describes a Multi-Agent System, including advanced features like cooperation and conflict resolution, controlling a flexible robotics cell, we have set up at the University of Porto as a testbed for Archon methodologies (UPART - University of Porto Assembly Robotics Testbed).

The first chapter introduces the motivation for our research, as well as this document structure. The second chapter aims to explain what are the main characteristics of the problem we have chosen, and why can it be seen as representative of a class of good examples for MAS architecture design.

Chapter three introduces our approach to the MAS architecture as well as Knowledge for the Agents themselves. We also emphasize the use of the Blackboard for intra (and not inter) agent communication, the enhancement of cooperative policies by the use of real priority management and very simple conflict resolution strategies.

A brief description of Agents involved in our testbed, as well as a simple scenario involving conflict resolution are included in chapter four. The last chapter is a short conclusion about the relevance of the research as well as experiments that have been done using the robotics testbed UPART.

2 - THE PROBLEM

2.1 CHARACTERIZATION AND DESCRIPTION

Flexibility is the key word in Intelligent Robotics. Limitations on the capabilities of current industrial robots need to be overcome in order to increase the number of application domains, mainly in which parts manipulation and assembly concerns.

The use of Symbolic Planning and sensors feedback on one hand and, on the other hand, the use of differentiated knowledge and skills in the robotics field lead to the following guide-lines of our work:

A- Symbolic Knowledge Representation and Problem Solving Reasoning Mechanisms are needed to control a robotics cell.

B- Time tagged information and actions, ought to be manipulated in real environments by appropriated time sensitive algorithms which include processes priority management.

C- Cooperation between different skills will increase the overall system capabilities to solve complex tasks.

2.1.1 General Description

UPART, the Robotics Testbed that has been developed at the University of Porto aims to be suited for assembly of units made up of different parts where millimetric precision is not required. Its main characteristic is to be flexible, therefore being able to perform several different assembly tasks with the minimum amount of reprogramming.

The layout for our testbed is composed of a simple table, 50 cm high, on the top of which some objects have been placed in order to be manipulated by a 5 and 1/2 degrees of freedom robot arm.

The robot arm, equipped with proximity sensors, is being controlled by a PC/AT running a robot level programming language called LM. Moreover, a Camera+Laser image acquisition process was developed in order to gather numeric and geometric data from each current layout.

Although all the computers are linked through ethernet, it is worth to notice that Computer Vision as well as Action control are based upon PC/ATs while all other AI oriented programs run under UNIX on SUN Workstations.

What we claim this testbed has been suited for, is the useful illustration of a Multi-Agent System architecture controlling a robotics cell providing facilities concurring for the solution of assembly robotics problems involving flexibility and efficiency.

2.2 THE EXPERTISE

Selecting a MAS architecture to deal with the problem described implies the identification of the needed expertise for solving the proper goals. Which Knowledge has to be available to the system to enable the joint activity of agents towards the completely automatic assembly and manipulation of different objects? We can identify, at least, the following types:

- Knowledge for **interpretation**
- Knowledge for **decision**
- Knowledge for **action**
- Knowledge for **coordination.**

Let us postpone description of Knowledge for coordination till chapter 3.

Scene interpretation asks for Knowledge on sensorial information acquisition, representation, classification and possible fusion in order to be able both to locate and to identify real objects in the real world.

Visual information as well as complementary distance

measurements have to be processed for - at least - objects identification. Other functional characteristics of objects related with their possible manipulation, like stable states and grasping positions are also mandatory whenever the task involves physical changes in the world.

Decision making usually follows the interpretation step. In the selected framework, the decision capability is neither a diagnostic nor a simple order but, on the contrary, is a more or less complex plan that has to be followed in order to coherently, and efficiently, change the world.

The knowledge involved in this process includes world symbolic representation. At the core of this knowledge for action are also planning algorithms which take into account all kinds of possible operations for the applications, motion interactions and restrictions.

Our robotics testbed is a good example of the effects of the MAS architecture on the action upon a restricted world. Actions much more efficiently performed, in a variety of situations, are the visible result of a set of primary decisions on Systems organization and architecture.

To know how and when to gather information as well as how to translate intentions on actions for object manipulation does not involve a trivial kind of knowledge - sometimes depending on particular robots and languages - that has to be taken into consideration.

All these kinds of knowledge for interpretation, decision and action have been divided and supported by six different semi-autonomous agents which work together to accomplish the proposed goals.

3 - A MULTI-AGENT SYSTEM ARCHITECTURE

3.1 AGENT ARCHITECTURE

The following Figure shows our Agent Blackboard Architecture. The domain related part of the Agent, called the Intelligent System (IS), may either be an Expert System, a Planner or even a more procedural program. What makes it to be seen as an Agent is the layer on top of it which is responsible for coordination (of both local and cooperative activity) and, therefore can be called the Coordination Layer. A more detailed discussion about Agent architecture for Archon has been discussed in [1].

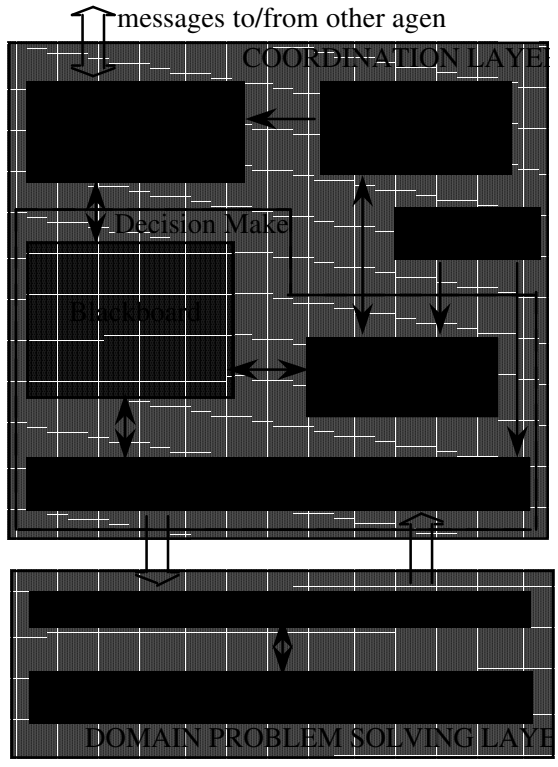

Fig.1 ARCHON Agent Blackboard implementation

In the Coordination Layer we can distinguish several modules:

- High Level Communication Module (HLCM) responsible for the communication through Unix message passing and capable of sending (broadcasting, multi-casting, point to point) as well as receiving messages from other Agents;
- Cooperation Module, responsible for coordinating cooperative activity (agent's social behavior). It contains knowledge sources for cooperation;
- Monitor, responsible for coordinate local activity (agent's individual behavior). It contains knowledge sources for I. S. monitoring;
- Agent Acquaintance Module (AAM), where all needed information about other Agents which are of interest for this particular one, resides;
- Self Model (SM), embedding an abstraction of the own Agent capabilities, its skills and interests;

In the framework of this implementation, the Decision Making Module, which can be considered as the heart of the Agent, is blackboard based, more precisely, it uses the GoalBlackboard/DataBlackboard approach for intra (and not inter) Agent communication. Knowledge that Decision Making Module uses either for Cooperation and IS Monitoring, is represented by means of Knowledge Sources

(KSs), set of rules that have been described elsewhere [2].

When and how to cooperate is an issue of great importance for an Archon agent. Each cooperative agent should be able to know when it is necessary to cooperate with the others, and this rises the problem of deciding **when to cooperate**. **How to cooperate**, i.e., what cooperation strategy to use in each particular situation, is another problem at a different level.

While Data Blackboard (Data-BB) is the place to put the messages coming from both the other agents, the role the Decision Maker plays in our framework determines the contents of the Goal Blackboard. Two kinds of control decisions have to be made: decisions on interacting with the others (e.g.. how to decompose the task, etc.) and the ones for local task scheduling (e.g.. what task the IS should do now, etc.). Therefore, Goal-BB encompasses goals to make these two kinds of decisions.

3.1.2 Knowledge for BB Decision Maker

As a reasoning module, the Decision Maker has a relatively simple control mechanism --- the three-step loop: Generate Goal, Select Goal and Execute Goal.

The key problem turns to be how to map the incoming messages into the changes on the GoalBB. Intuitively, these changes are made by the set of Knowledge Sources (KSs) linking the two blackboards at the Generate Goal step. Triggering one of them may cause changes on the GoalBB (the generation of a new goal, or the state change of a goal, or the change of a goal's priority).

At the Select Goal step, the Decision Maker chooses the most appropriate goal in terms of goal's priority and state. This chosen goal will then, at the Execute Goal step, trigger the corresponding KS to execute it. These three steps obviously represent three different kinds of actions related with decision making: intentions generation, situation assessment and execution of intention.

3.2 COORDINATION ACTIVITIES

Decision Maker has two different sets of Knowledge Sources (KSs) corresponding respectively to cooperative activity and local activity coordination.

These two sets of KSs can communicate with each other via DataBB. As an example, when the agent has decided (by means of the cooperation KSs) to execute a task so as to help the others, it will generate an event on the DataBB which, as a consequence, will trigger a task-scheduling KS to generate a goal on the GoalBB enforcing IS to do it.

In the following, we will discuss how the Decision Maker deals with these two kinds of reasoning activities within the same uniform mechanism.

3.2.1 SUPPORTING COOPERATIVE ACTIVITY

Cooperation KSs have three kinds of abilities: **detection** of need for cooperation, **choice** of Cooperation Type and **execution** of chosen cooperation policy. This set of KSs is then the primary responsible for the agent's cooperation behavior. Another set of KSs is intended for the purpose of controlling the underlying IS and, therefore, it determines the agent's local problem solving behavior.

In which Cooperation concerns, Agents can play two main different roles: They can be seen as **Organizers** of the Cooperation or as **Respondents** to other Agents. Knowledge Sources are available for the Organizer either to distribute tasks (Task Sharing) or to get several results from the other Agents (Result Sharing).

3.2.1.1 Cooperation through Negotiation

If there is only one Agent who has the capability to answer to the Organizer's request, then it generates a goal in the Blackboard to initiate a Client/Server relationship. In the case that more than one Respondent is able to execute the task, then Organizer generates a goal to initiate Contract Net protocol.

To be able to choose among possible cooperation demands which one to answer, the respondent also needs some specific knowledge. In order to execute the selected cooperation protocol and eventually establish the cooperation relation, each cooperation protocol must be complete. Therefore, in addition to the task-announcement KS, the following KSs are needed for contract net:

- bid-KS triggered by a bid message, it evaluates bids and sends the appropriate award.
- award-KS triggered by award message, it is responsible by task decomposition and to send task announcement message.
- termination-KS triggered by termination message, terminates execution of the contract and all related subcontracts.

3.2.1.2 Cooperation and Conflicts

Negotiation for Cooperation is a very important capability Agents shall exhibit, but Conflicts may appear

exactly at the very core of the Cooperation process. Whenever an agent wants to formulate a request, in order to get an answer from other agents, it needs to establish a protocol with the possible respondents for that specific request. This protocol will enable the exchange of information between the agent interested in getting the answer and the possible respondents (which may have to execute an IS task to provide the answer to that request). During this protocol for message exchanging, the Organizer informs other agents about the constraints related to that request (mainly a deadline). The potential contracted agents will respond informing about the expected time needed for having that task executed and, probably, the expected quality of the answer (bid message). The Organizer will face then a positive Conflict to be resolved. A selection of the appropriate agent has to be made.

Therefore, **Conflicts** are always possible to happen during a Cooperative Multi-Agent Community activity. A broader classification of Cooperative and Conflict situations is presented in [3]. Strictly speaking, conflicts may appear either during Task Sharing or Result Sharing operations and may also be classified either as positive or negative. Therefore we may recognize the following types of conflicts:

- positive conflict in task sharing: whenever several agents are able to execute a task required by the same organizer. This situation has to be tackled down by means of negotiation functionality's
- negative conflict in task sharing: whenever there are no agents willing (or able) to execute a task required by the organizer. For real time system we can extend the concept to the impossibility of giving the desired results in due time. This situation requires a new plan (if available) to be issued by the Organizer's Monitor in order to find out a new and feasible plan for that specific task
- positive conflict in result sharing: whenever several agents, which are trying to execute the same task, produce different but complementary results for the same request (or the same result with different credibility messages associated)
- negative conflict in result sharing: when several agents, who are executing the same task, produce either inconsistent or antagonistic results. In the later case, we have different alternative answers to the same request (they can also be seen as different values for the same receiver's task input), and, in the former case, there are inconsistencies detected between different (but somewhat related) information which are inputs to that task. The capability to deal with these situations and to resolve the conflicts rely mostly on the availability of a functionality to measure information quality.

Whenever no Agent is able to satisfactorily respond to another ones request, the Organizer must either accept the unsatisfactory answer (e.g. later than desired) or change plans for the Task it has in hands. This policy does not require too much complication. On the other hand, real conflicts are those which happen during Result Sharing. When several results for the same request are available there is a need for domain dependent knowledge specifically appropriated for the resolution of the conflict. This fact had led us to propose the assignment, for each type of conflict information (results of Task T_j, a specific agent Ag_{crj} in charge of the resolution of that kind of Conflict. Therefore, whenever an Agent Ag_i has an information (possibly a result of one of its own tasks) for another one, and it knows there is a third one (possibly more) trying to answer the same request, it feels obliged to send its results to the agent $Ag_{crj.}$ This Ag_{crj} is specified in the AAM of Ag_i as an expert on resolving conflicts regarding that information Ag_i is about to produce.

It is up to agent Ag_{crj} to resolve the conflict and send the result to the Agent which had made the primitive request. Procedures for resolving specific conflicts have to be available at each agent Ag_{crj} level.

In the Decision Making Module there are sets of rules which are activated when Conflicts between Agents have to be resolved. These Rules include generic Knowledge, hence are not application dependent. Moreover, in order to be effective, the high level knowledge involved in these rules must rely on other specific lower level enabling functions including domain related actions.

Reasoning about Conflict Resolution using appropriate Knowledge also implies, as it was the case for Negotiation, the use of some lower level enabling functions which cannot be described here.

3.2.2 AGENT'S LOCAL ACTIVITY

We have identified four kinds of possibilities for task scheduling of the Intelligent System, other than normal task initialization:

• task swapping : During IS's execution of a task, whenever an urgent request (task) arrives, the Decision Maker controls the IS in order to suspend the current execution and to swap to execute the new arriving task. When the new task has been concluded, the IS then should continue the previous execution from the breakpoint.

• restart task : During IS's execution of a task, if more reliable data which are the partial or full input to that task arrives, the decision maker should control the IS to restart the task's execution.

• redo task : If more reliable data which are the input to an already executed task arrives, the decision maker should control the IS to redo that task and control HLCM to send the newly produced results to the other agents who are interested in.

• multi-task execution : If the underlying IS has the ability to do several tasks simultaneously, the decision maker should be capable of monitoring the executions of several tasks at the same time.

Redoing an executed task or restarting a task under execution with more reliable data can guarantee that each agent's contribution to the global task is positive. Task swapping gives the agent the capability to process the task in the due time. Multi-task execution can surely reduce the total amount of time the whole system need to finish the global task.

3.3 MANAGEMENT OF PRIORITIES

Temporal Deadlines are of great importance for those applications where the produced answers to a certain request are only useful if they are available until a specific point in time (real time constraints). However, deadlines may be used to improve the efficiency of the system performance paying more or less attention to specific tasks, varying their task priorities according to their relative importance or urgency. Therefore, to guarantee the accomplishment of a specific task according to a deadline, an Agent shall compute *Priorities* taking also as much as possible into account information about the current system situation. Although requests are formulated with specific temporal deadlines attached, and not priorities, it is the receiving agent job to compute the more appropriate priority value for task scheduling purposes.

A complete functionality to calculate appropriate Unix priorities for Agents tasks trying to commit Agents to their tasks deadlines, is also part of the knowledge each Agent has available.

4 - ROBOTICS TESTBED

4.1 AGENTS DESCRIPTION

A more detailed description of the Agents involved in the UP Robotics Testbed has been made elsewhere [4] [5]. In this section we will only briefly describe the main purpose of the 6 Intelligent sub-systems integrating the Testbed. The Agents are called (from their Intelligent System functionality):

TLP - Task Level Planner - a high level non-linear plan generator

ELP - Execution Level Planner - responsible for executing the plan and table space management,

VISION - Object Recognition System based on Computer Vision

LASER - 3D Object Recognition System based on Range Finding Techniques

MODELS - Data Base representing object features important both for identification and manipulation

WD - World Descriptor - for extracting objects symbolic relationships and constraints

4.2 EXAMPLE

This sub-section will show an example running in our Robotics Testbed, which tries to demonstrate some of the functionalities already described in previous sections.

In order to execute his own goal - *execute_assembly_task* -, ELP needs to know the current location of the objects - *object_locations*. This information can be provided both by VISION and LASER. ELP will start a negotiation process, in which it will play the role of Organizer. It will send an announcement message to both Respondents, asking for that information with an attached deadline - *Deadline* :

announcement(object_locations,Deadline)

From : ELP To : VISION and LASER

LASER and VISION Agents will respond with a bid message, specifying their current availability. Agent's availability includes the expected time to conclude a task that will produce the required information - *ExpTime* -, and the number of inputs needed for that task which are not available at the moment -*InpNotAvail*.

bid(object_locations,ExpTime,InpNotAvail).

From : VISION and LASER To : ELP

ELP will evaluate all bid messages and select the most appropriate Respondent agent (suppose VISION), to whom a request message is to be sent:

request(object_locations,Deadline)

From : ELP To : VISION

VISION will now execute the task *get_objects_locations*, in order to provide the required info. Meanwhile, LASER will also execute the task *get_object_locations* and produce the same information, because he is idle and all inputs needed for that task are available. When LASER concludes the execution of its task, it looks at AAM and sees that VISION is an expert on resolving conflicts regarding the information *object_locations*. LASER sends then its results to VISION:

answer(object_locations,Laser_Results,laser).

From : LASER To : VISION

VISION will now combine LASER and its own results related to the information *object_locations*, and by activating the appropriated conflict resolution procedures will produce a final result. This situation is illustrated in figure 2.

5 - CONCLUSIONS

UPART, a Robotics Testbed we have implemented at University of Porto, gave us the possibility for experimenting "in situ" most of the ideas related with the Multi-Agent System architecture as well as cooperation strategies Archon consortium has developed.

The Multi-Agent structure, the granularity and degree of autonomy of Agents, their knowledge organization have proved to be suitable for the kind of problems we wanted to look at in the field of assembly robotics.

On the other hand, the Testbed itself, induced some new problems derived from the fact that time was an important consideration for the robot action. Inclusion of deadlines and priority management was also a contribution the robotics field gave to the coordination policies Multi-Agent Systems have to take into account.

All the Agents have been implemented in our Lab and every functionality we described here can be demonstrated. Informal measurements concerning execution time to perform several tasks both with the Multi-Agent approach and the hierarchical one has been made leading us to the obvious conclusion that the Decentralized Multi-Agent approach is not only more natural but also more efficient than the traditional one.

ACKNOWLEDGMENTS

I am deeply grateful to all members of AI&Robotics Group at Porto University, who have contributed to the implementation of the ideas here presented. Special thanks are due to C.Ramos who, during his Ph.D. work has implemented some of the Agents.

BIBLIOGRAPHY

[1] T. Wittig (Editor), "Archon: Architecture for Cooperative Multi-Agent Systems", Ellis Horwood 1992

[2] E. Oliveira, F. Mouta, C. Ramos, A. Rocha, "Conflicts and Cooperation Analysis within ARCHON", UP/Tech. Note 14 of ARCHON Project, 1992

[3] E.Oliveira, F.Mouta, A.Rocha, "Negotiation and Conflict Resolution within a Community of Cooperative Agents", Int. Symposium on Autonomous Decentralized Agents, Kawasaki, 1993

[4] C. Ramos,E. Oliveira, "Closing the loop of task planning, action and sensing", IEEE/RSJ-Int. Conf. on Intelligent Robots and Systems, Raleigh (USA), 1992

[5] C. Ramos,E. Oliveira, "Sensor Based Reactive Planning and Execution for Assembly Tasks", IEEE- Int. Conf. on Systems Engineering, Kobe (Japan), 1992