# A multi-agent environment in robotics

Eugénio Oliveira, R. Camacho and C. Ramos

**SUMMARY**

The use of Multi-Agent Systems as a Distributed AI paradigm for Robotics is the principal aim of our present work. In this paper we consider the needed concepts and a suitable architecture for a set of Agents in order to make it possible for them to cooperate in solving non-trivial tasks.

Agents are sets of different software modules, each one implementing a function required for cooperation. A Monitor, an Acquaintance and Self-knowledge Modules, an Agenda and an Input queue, on the top of each Intelligent System, are fundamental modules that guarantee the process of cooperation, while the overall aim is devoted to the community of cooperative Agents. These Agents, which our testbed concerns, include Vision, Planner, World Model and the Robot itself.

KEYWORDS: Multi-Agents; Cooperation; Knowledge; Robotics; Distributed *AI*.

## 1. INTRODUCTION

The principal aim of our research in the *D*istributed *A*rtificial *I*ntelligence (*DAI*) field, using the Multi-Agent Systems paradigm, is to design a suitable system architecture for the coordination of different intelligent skills, specific to each one of various semi-autonomous agents, inside a global and coherent cooperative community exhibiting intelligent behavior when facing a complex task. Our testbed is a flexible assembly robotic cell using Vision and other different sensors and being commanded by a User in a high-level task oriented language.

An architecture of a decentralized set of active units which are able to exchange many different types of information in multiple directions may be constructed by different semi-autonomous Agents, each one with a certain degree of intelligence.

In order to be considered as semi-autonomous, an Agent must have its own problem-solving capabilities. Some default reasoning is also needed to avoid the case of being idle whenever the requested information is missing when required. It is not desirable that an Agent should remain in a waiting state after a request has been made. Such Agent shall focus its attention on another problem, and from time to time check if the answer has already arrived.

Complex problems may need the concurrence of several skills, competences and abilities in order to reach an acceptable solution. This does not necessarily imply a need of a pre-defined splitting of a problem into sub-problems and the attachment to each computing actor of one of these sub-problems, while waiting for all the partial solutions in order to fuse them into a final one.

A richer form of cooperation goes beyond a rigid problem splitting or task-sharing, and may imply "simultaneous" efforts to answer the same question by more than one Agent. This involves a result-sharing facility enhancing the cooperative aspects of the community.

The main idea of a community of Agents that are semi-autonomous is to propose an architecture where these Agents can go on trying to solve the problem but, at the same time, are able to put questions to each other, to accept their answers and to progress with them, in order to get a better solution faster.

A collection of semi-autonomous Agents will then run in "parallel"* their problem-solving capabilities and cooperate via messages communication facilities.

In a cooperative "community", Agents are supposed to formulate queries, to receive specific requests from others and to accept useful information (answers to their questions or voluntary information from another agent). Therefore, cooperation is seen as a much richer concept than communication. In fact, what is involved it is not only the exchange of data, but also the possibility to assign other tasks to the Agents and redirect the flow of computation according to the incoming information. The paradigms of *D*istributed *A*rtificial *I*ntelligence[1-3] seem to be adequate to solve the problem of cooperation among different Agents whenever knowledge transfer is needed.

## 2. MOTIVATION FOR COOPERATION IN ROBOTICS

Whenever a robot arm is faced with a non-trivial task like the assembly of parts which are present in a incompletely structured environment, two main factors should be taken into account which suggest a **DAI and a Multi Agents approach as paradigms.** These factors are:

- Diversity of expertise required at different levels.
- Set of constraints under which these Agents must perform, therefore cooperate, to solve a problem. Moreover, the inherently splitting of many tasks, is

* The concept of parallelism should be substituted by the one of interleaving, whenever different Agents run in the same machine.

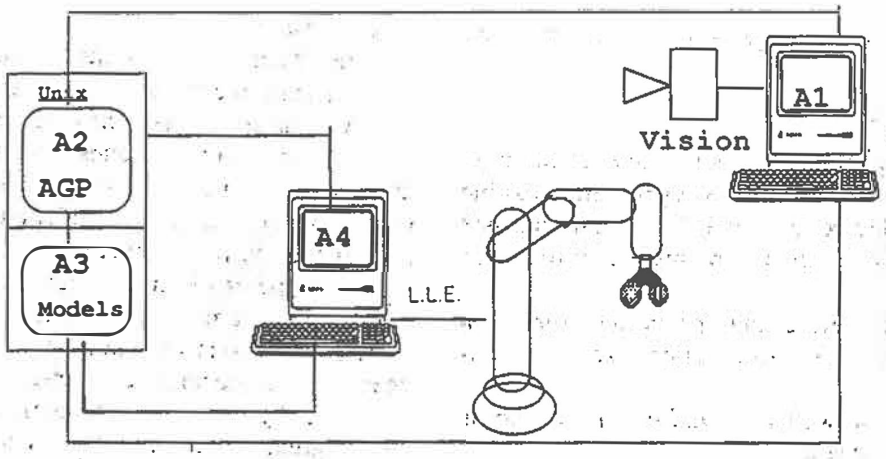


Fig. 1. Four Agents for one robotics cell.
A1: Objects Recognition
A2: Automatic Generation of Plans
A3: Model of Objects and State of the world
A4: Trajectory planer and Sensor Monitoring

another factor pointing to the potential benefits of close cooperation among different specific Agents.

Each one of the given tasks to be executed in an incompletely determined robotic environment, implies planning by an intelligent system of the work to be done. Such kind of "intelligence" needs, in order to be effective to gather numerous significant inputs from the world-either geometrical or symbolic information; which is made available not only by computer Vision, but also by *a priori* knowledge about possible expectations (Models).

Moreover, an accurate action performed by a physical device, like a robot arm, has to be supervised at each step by special agents monitoring its actions and correcting on-line, its deviations.

Therefore, it is obvious, that a community of cooperating Agents provides a possible and elegant solution for a flexible robotic workcell design.

## 3. COOPERATIVE COMMUNITY OF AGENTS

The community of Agents we have implemented, is made up of units which are not antagonistic but complementary. This does not mean that there is no competition among such Agents. In fact, due to a certain degree of overlapping expertise, one among several candidates may be chosen to answer a question or solve a sub-problem at a certain time.

As an example, we can imagine that whenever a gripper loses an object during movement, either a set of proximity sensors plus low level planning or the vision Agent, plus high level planning Agents, may try to command the recovery of that lost object.

In our approach we are playing with Agents which are to a certain extent semi-autonomous. This means that if some requested information is not obtained or is missing in useful time, an Agent must use its own fault-reasoning capabilities in order to avoid the situation of being idle. Therefore they may guarantee their own degree of "intelligence" to obtain a possibly poor solution even if they don't get convenient help from others.

Nevertheless, the principal aim of the community is to exchange knowledge and data among Agents in order to solve some kind of assembly tasks under more or less severe constraints. This cooperation takes place in a physical environment, using communication links between different processes or machines. The definition of protocols, both at a knowledge level as well as at communication level, is then crucial.

## 4. AGENTS

An Agent is an active computational unit with its own problem-solving capabilities, able to be executed separately and trying to solve a task the best it can. Moreover, it has the "perception" of being a part of a bigger "community" of Agents, with different skills and objectives, nevertheless, all of them are interested in cooperating to obtain better solutions to the problem under consideration.

Each Agent has its own inference mechanism, knowledge (or just data if it is not an "intelligent" one) and communication ability. It can sense the world and exchange data, knowledge, goals, part of plans, beliefs (with associated measures of uncertainty) and commands.

In order to do this, an Agent must know the profile of its surrounding community, which implies a stereotyped representation of different and relevant aspects of the other Agents.

Moreover, their own self-knowledge (to know the contents and limits of their own skills and competences) enables them to select other Agents for the sake of a

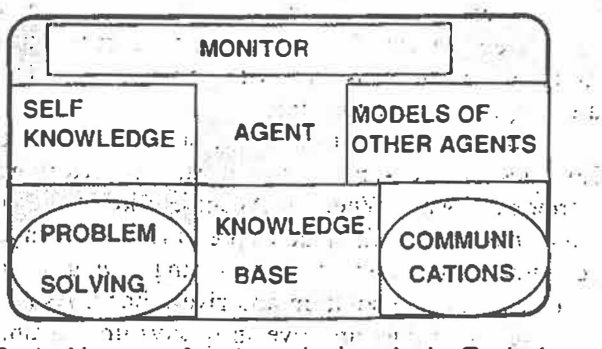


Fig. 2. Architecture of an Agent in the robotics Testbed.

fruitful cooperation, sending queries to and accepting requests from them.

## 5. ACQUAINTANCES

Semi-autonomous Agents in a cooperative community need to know each other, at least as regards the possible benefits of the intended cooperation. The question arises how shall the images of Agents $A_i$ be presented to Agent $A_k$?

There have been interesting proposals for such acquaintances models[2]; we will summarize these, as follows:

–*Name* of Agent $A_i$ as well as an *address* in order to be recognized and to be located;

–Agent's possible *role*. Is it a creator of other Agents or just a member of the organization.

–*Skills*. The Agent's competence and domain knowledge.

–*Goals*. List of goals other Agents know that one is able to achieve.

–*Plans*. Some information about *how* each specific Agent can reach a conclusion.

Therefore, a specific Agent $A_i$ can be activated, selected, or required by other $A_k$ on the basis of any of the above-mentioned characteristics.

It is also our opinion that each specific Agent shall contain their own models of its acquaintances, avoiding the existence of a separate entity with all that knowledge about the community; the latter possibility could indeed, lead to a bottleneck during cooperation.

Another important aspect of the intended modelization of the "community" is the inclusion of Model(s) of human Users which almost inevitably will interact with the system. It is therefore important to build up User Models in order to improve the human-computer Interface. This matter of User Models in the Agents Community is the subject of another on-going research taking place in our group. Satisfactory approaches to User Modelling can also be found in Walhster and Kobsa.[4]

## 6. STRATEGIES FOR REASONING DURING THE COOPERATION

Since they are autonomous and intelligent, each Agent must be provided with its own reasoning mechanism. Different Agents may use different reasoning methods. Therefore the distributed system is composed of heterogeneous modules and their reasoning methods may depend on the domain of expertise. In such a distributed system there may exist heuristic intelligent strategies, but it is also likely to find non-*AI* systems that use simply coded algorithms.

In time-critical environments, such as is the case in Robotics, the reasoning process can be performed in an **incremental** way. When an Agent is asked for a solution it may have a default value to present, if necessary, to the demander and then start reasoning trying to quickly achieve a possible solution and give it back. As time goes by, given solutions improve in quality until a time limit has been reached.

### *a. Uncertainty*

The heterogeneity of Agents may not only be present in the reasoning methods but also in the methodologies to deal with uncertainties. These can range from numerical methods to symbolic ones. Numerical methods can directly use either Bayes' theorem, Certainty Factors model or be based on evidential theory (Dempster-Shaffer theory, for instance).

In a cooperative environment each Agent can obtain information from other Agents using different representation methods for uncertainty. Therefore there is a need to transform such representations so that the receiving Agent is able to admit them. Such facility is also of great use in a robotic environment.

As an example, let us suppose that the Agent in charge of modeling the world has a measure of uncertainty using intervals to deal with preconditions to grasp an object. If the location of that specific object has been extracted by the Vision Agent with a single figure to represent the uncertainty of that location, a process of matching between the two information sets should be performed, despite the referred difference in the representation formalisms. Simple and somehow reductionist techniques are being investigated to find a number in the interval in the first representation to be matched with the figure obtained on the second representation.

### *b. Truth maintenance*

The need of *TMS* or *ATMS*[5] arises when different (eventually contradictory) information sets are generated in a system during information gathering or in the process of inference. As we have mentioned earlier concerning incremental reasoning, there may arrive to each Agent different answers for the same problem at different stages of the inference process. This implies the need to have a sort of a direct-dependency network with previous conclusions which may be easily updated.

We can deal with this non-monotonic aspect of knowledge using either *TMS* (Truth Maintenance Systems) or *ATMS* (Assumptions-based Truth Maintenance Systems).

In a Robotics environment the need for *ATMS* may happen, for example, when the Vision Agent gives erroneous or imprecise information to the planner. When the latter Agent starts reasoning with that information, planning some paths or grasping actions, the Perception (Laser) Agent may give to it information that contradicts the one given by the Vision Agent. If the Acquaintance Model states that more credibility shall be given to the laser, this implies that actions already started must be revised. A global coherent image of the "world" is then built up by the *ATMS*.

## 7. EXAMPLE OF COOPERATION IN ASSEMBLY ROBOTICS

In this section we specify a testbed for cooperation in a Robotic environment. Some situations where cooperation is desired are characterized, and specific Agents for assembly robotics are presented as well as a very simple example of cooperation.

*a. Requirements for cooperation*

Each Agent is seen as a set of different modules, controlled by its own Scheduler (Monitor), and supported by different processes running under Unix. This Scheduler (Monitor) is able to manage the following modules: An input queue, where the requests to each specific Agent are situated; an Agenda where different sub-tasks to be performed are kept; and the double aspect of knowledge (Self-knowledge and knowledge about the Acquaintances).

Moreover, this upper layer, which is responsible for cooperation, is also in charge of controlling some aspects of the underlying Intelligent System according to the requests of the cooperative community (see Figure 3).

Agent queries and answers are transmitted by sending messages to the in-queue of the receiver Agent which is selected according to indications obtained from the Acquaintance module. The Monitor takes the received message, analyses it and, depending on its type (a goal to be executed, a fact to be inputed or a plan to be performed), places it in the Agenda waiting to be considered by the Agent's Intelligent System.

Messages format includes, other than the content, the sender identification, the message type and possible time constraints. On the other hand, the Monitor of the receiver agent handles messages and assigns priorities to them, according to their type, the importance of the sender and the status of the Agenda of the receiver.

The possibility to redirect the requests is another important feature of the Monitor. An Agent must have the opportunity to interrogate a set of Agents. The number of elements of this set may be one, some or all other Agents of the community (broadcasting). However, whenever questions are formulated, the Agent should be able to go on "thinking"about other problems, while waiting for answers from others Agents. Moreover the agent needs to verify, whenever desirable, if the requested answer has already arrived. Some problems can occur if the Agent remains in a waiting state because the desired answer cannot become available. Attention is also paid to prevent situations where a circular linking of requests leads to a "cooperative deadlock".

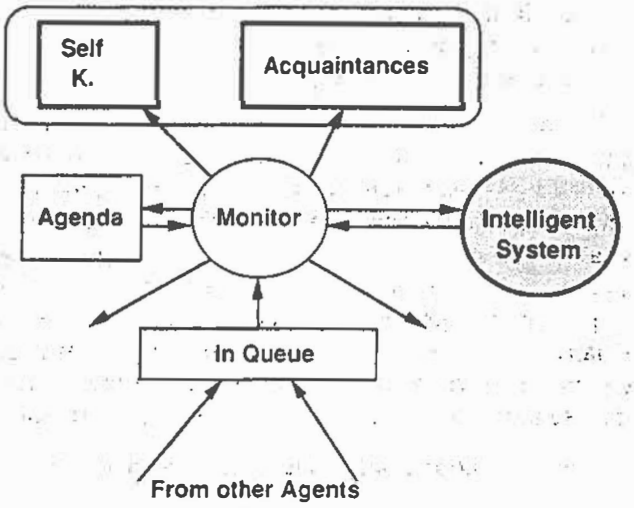


Fig. 3. Flow of information inside an Agent.

In conclusion, we notice that important cooperative features, like default reasoning, broadcasting and requests to specific Agents must also be present in a robotic testbed controlled by a community of Agents. Nevertheless, we shall see that default reasoning has poor credibility in this case due to the highly dynamic environment and the irreversible consequences of previous actions. This implies a real need for planning activities and correct agenda creation by the cooperative layer.

*b. Cooperation needs for the robotics workcell*

In the framework of our robotics workcell it is possible to identify the need for several Agents, such as Planners (High-level and low-level actions), Vision Interpreter, other sensors (laser, proximity sensors, etc...) and the World representation Agent. Below, some situations are depicted where cooperation between Agents (namely Planning, Vision and Object Models + World representation Agents) is desirable:

(i) Whenever the Planner knows *a priori* which objects are present in a specific scene, but not where they are, symbolic objects features (like name, function, color, weight, material etc...) may be sent to the Model representation Agent. This Agent associates symbolic features with numerical dimensions of the modeled object and send this information to Vision. This definitively improves the objects search strategy, because Vision looks only for the desired objects on the image, ignoring other information on the image.

(ii) On the other hand, if a Planner completely ignores which objects are present on the scene, Vision starts to work upon the scene identifying objects and sending their identification to the World descriptor Agent and to the Planner in order to enable their action. Let us suppose that the Planner does not yet know the task to be performed (the cell is able to perform several tasks) and Vision identifies one object on the present scene. This may be enough for the Planner to start with a planning task. This event triggers, by itself, a new request to Vision in order to be informed about locations of other specific expected objects.

(iii) There are also situations where time is a critical constraint and planning tasks impose on Vision a quicker answer. If this is the case, the Vision process must speed up applying faster processing (using, for example, the Hough Transform for only one pixel by each 4 or 16). Clearly, such an improvement leads to quicker answers despite some loss of accuracy. Nevertheless, nothing prevents Vision to go on elaborating a better solution.

(iv) Contingency analysis or exception handling (what to do when unexpected obstacles are met or a loose object) is still another field where cooperation may be of great importance.

*c. Agents in our testbed*

As regards the description of the cooperation principles in a robotics environment, we have already developed

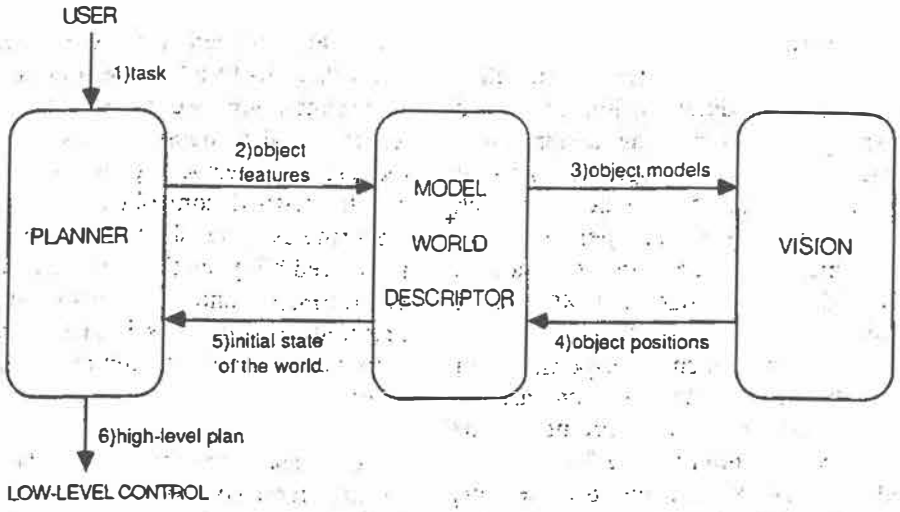


Fig. 4. Cooperation among three Agents.

three main cooperative Agents:

- Vision Agent
- Planning Agent
- Model + World Descriptor Agent

Our vision Agent is a simple 3D object identifier using one image (software to extract a matrix of distances from Laser information is under development) and some *a priori* knowledge about objects to locate (numeric model of objects). The camera positions (focus point, focal length, image plane equation etc...) relative to the space axes $(x,y,z)$ are assumed to be known.

Vision processing starts with the Hough Transform application to detect lines.[6] It uses a methodology known as "Preview, Verify and Validate" in order to recognize objects with the help of appropriated Models.

The High level Planning Agent obtains from the User the description of the task to be done (including the objects to be manipulated characterized by their attributes). For example, a specific task may be presented as follows: is_goal(put_on(small_cube,orange_object)).

Positions of the objects and the complete description of the state of the world are provided by the user, or automatically gathered from another Agent. For the purpose of our description about the cooperative methods employed in our testbed, we only describe here the above referred three Agents. However, it is clear that other Agents of a lower level nature, like the robot controller and the set of sensors for monitoring, are mandatory in the robotics testbed. Problems involved with execution of actions in the real world raise other kind of problems, like real-time responsiveness, or grasp planning that are beyond the scope of this paper. Nevertheless, the fact that the real testbed encompasses a more complete set of agents should be kept in mind.

Another Agent is responsible for object models and world representation. The object models are represented as frames with slots reserved for each symbolic or numeric feature. For example:

| | |
|---|---|
| number: | 3 |
| name: | box1 |
| material: | plastic |
| color: | yellow |
| weight: | 100 |
| size: | small |

numeric charact.: perimeter, surface, center of mass . . .
position: ⟨coordinates⟩

The numeric dimensions of the object are also defined. Other interesting features to be considered are possible grasp positions for each object depending on specific tasks. Moreover, World representation also uses position coordinates (obtained by Vision) to infer some symbolic positional relationships of objects in the world.

*d. A simple example of cooperation*

In Figure 4 an example of cooperation between the three Agents described before is represented.

This simple example concerns a situation with three different boxes whose image has been acquired and processed:

a white box
an orange plastic box
a small red wood box

Goals are specified by the user as follows:

is_goal(on(white,floor)).
is_goal(on(plastic,white)).
is_goal(on(small,plastic)).

The objects attributes as white, plastic and small have different natures (color, material and size). A message with those goals is sent to the Agent containing Models which uses the aforementioned attributes (white, plastic, small . . .) to find the appropriated object model. Geometrical information is found and given to Vision, helping in the process of locating each specific object. Finally, the World Model uses the object positions to obtain the description of the initial state of the world and then sending to the Planner the following information:

(on(white,floor). on(plastic,floor). on(small,white).

Now the Planner can establish the high-level plan to

execute the task:

put_on(box1(small),floor).
put_on(box2(plastic),box3(white)).
put_on(box1(small),box2(plastic)).

Since our laboratory testbed is equipped with a robotic system (Renault/APRA) controlled by a PC/AT (MSDOS) using *LM Language* programs, the previous example has been executed after transferring object positions and the derived plan to PC files. A very simple low-level planner (on the PC) has successfully executed the planned task.

This low-level planner, is not only able to translate high-level primitives into *LM* procedures, but also inserts some monitoring actions along the critical points of the program. This allows a finer control of the execution of the final robot movements.

*e. Self-knowledge*

We have seen before that cooperation between Agents implies the existence of two main kinds of different pieces of knowledge: Self-knowledge – To know each own Agent capability – and, Acquaintances knowledge – to know other Agents' capabilities. In this section we go into further details concerning the contents of knowledge in the upper-layer capability of Agents that are specific to our robotic testbed.

Each specific agent of the cooperative community knows which goals it is able to execute. We can divide all possible executable goals and subgoals into two main categories:

Those which are the exclusive competence of that Agent (independent goals)

and Those which depend on other ones (dependent).

An independent goal or subgoal can be immediately executed by the Agent without any need of external information. On the other hand, if the agent needs external information on goal or subgoal is said to be dependent.

The structure, of Self-knowledge in which concerns goals, subgoals, external information (Data plus Knowledge) and the produced information (Data plus Knowledge) are then closely interrelated in the respective module. It is up to the Monitor to manage such rich information in order to drive the incoming requests and to plan the problem-solving activity of the receiving Agent.

The previous example, illustrated in Figure 4 (playing with three agents) is considered again in order to describe Self-knowledge involved in cooperation.

**(i) Self-knowledge for vision agent**

IS→ Object Identifier
Goal 1→ To *locate* objects for the given task
subgoal 1.1 – to obtain object models
input: geometrical models
from: (Object Models + World Model)
subgoal 1.2 – to acquire and process the image(s)
input: request
from: (Planner, User)
subgoal 1.3 – to generate and to verify hypothesis
*Result*: *spatial positions of located objects*
Goal 2→ To *verify* object positions
subgoal 2.1 – to obtain object models
input: geometrical models
from: (Object Models + World Model)
subgoal 2.2 – to obtain the desired positions
input: coordinates plus orientation
from: (Object Models + World Model, User)
subgoal 2.3 – to acquire and process the image(s)
input: request
from: (Planner, User)
subgoal 2.4 – to match Models with images
*Result*: *y/n*

**(ii) Self-knowledge for planning agent**

IS→ High-level Planner
Goal 1→ to *generate* the high-level plan
subgoal 1.1 – to know which objects are present in the scene
input: object symbolic features
from: (User, Vision)
subgoal 1.2 – to know the state of the World
input: object geometrical relationships
from: (Object Models + World Model, User)
subgoal 1.3 – to know the desired objectives
input: final state of the World
from: (User)
subgoal 1.4 – to build up the plan
*Result*: *A plan composed of high-level orders*
Goal 2→ to *replan*
subgoal 2.1 – to know the actual state of the World
input: object relationships
from: (Object Models + World Model, User)
subgoal 2.2 – to know if the objectives are modified
input: final state of the World
from: (User)
subgoal 2.3 – to patch up the initial plan
*Result*: *high-level orders to be inserted in the initial plan*

**(iii) Self-knowledge for models + world description agent**

IS→ Set of Models and Spatial Reasoning capabilities
Goal 1→ to access object models by some features
subgoal 1.1 – to collect object symbolic features
input: Object name (name,color,material . . )
from: (User,Planner)
subgoal 1.2 – to access frames to obtain geometrical models
*Result*: *geometrical models*
Goal 2→ to describe the state of the World
subgoal 2.1 – to collect object positions
input: object positions
from: (User, Vision)
subgoal 2.2 – to apply spatial reasoning techniques
*Result*: *geometrical and positional relationships between objects*

The field "from" (from where the information comes), referred above, is related with the Acquaintances described below. The Monitor shall be able to associate the field "input" with the appropriated acquaintances in order to identify the set of agents that must be requested.

*f. Acquaintances*

In the Self-Knowledge Module, an agent has information about its own capabilities in terms of possible goals it can achieve. But it also must know who is able to answer its needs and which data or knowledge will be required from itself.

As regards goals, subgoals, Data + Knowledge and results, the acquaintances are not just the union of all pieces of knowledge about the other agents of the community. Only the direct interactions with each particular agent have to be represented.

For example, the high-level Planner knows that the Models + World Descriptor agent are able to provide the symbolic state of the World (see item 5 of Figure 4), but ignores the fact that the last one will ask Vision about object positions to solve that request (4 of the same figure). In such a case, it is more important to the Planner to know that it is convenient to send symbolic features of objects like name or color (2 of the same figure).

As an example, what is needed for the Acquaintances Model of Models + World Descriptor is:
from Vision

"it is able to obtain object positions"
from Planner

"it is able to know symbolic features of objects"

This simplified description of other Agents will be enough to help in the cooperative execution of tasks by the community of Agents.

*g. The monitor*

We have seen before how the upper layer of our Agents provides knowledge about the cooperative community (other Agents), as well as about its own Intelligent System. The control part of this cooperative layer will now be described; this control part is termed the Monitor.

The Monitor is in charge of:

- To direct requests to and from the Intelligent System (IS);
- To inform other Agents of its own IS activities;
- To interrupt an IS current task and, later on, to resume it dynamically.

In order to be able to have such features the Monitor consults AAM about interests of what is being executed by the IS. Moreover, it consults the IS Module for the plan of execution of each task (a set of sub-goals) and it interrupts it whenever another request with higher priority arrives. The swapping is performed after a command to save the current computational context given by the monitor.

## 8. SOME EXPERIMENTS

*a. Initial considerations*

Experiments of Cooperation using the Cooperation Layer have been made considering the following Intelligent Systems:

HLP – High-level Planner
LLP – Low-level Planner
WD – World Description
MODELS – Models of Objects
VISION – Object Identifier

In the User console six windows are open, viz. five for the above Intelligent Systems and another for specific USER interface. Whenever a name of an Agent is given to be part of the overall cooperative community, three files are loaded: The Agent's Self-Model, Agents Community Model and the Intelligent System itself. Monitor and Acquaintances Modules are identical for any agent. On the other hand, Self-Model and Community Models have different contents for each agent, despite the similarity in its frame structure.

The first activity of any agent is to open its own message queue by looking to its own Self-Model where the name of the Agent's Message Queue is. The Monitor starts then looking for tasks by periodically accessing this in-queue. Each Agent is a set of two Unix processes running concurrently, one for the IS and another for its cooperative layer.

*b. The test problem*

In this test problem objects shown in Figure 5 are supposed to be modelled in the MODELS Agent.

Figure 6a represents the initial state of objects and Figure 6b the goal state.

The overall goal of this test problem is the generation of a high-level plan to go from the initial state to the specified goal state.

*c. Example of cooperation*

The cooperative system starts to work when the USER writes the following goal:

required_goal(generate_high_level_plan,[usr],300,new)

This acts as a request which is directed to the HLP message queue.

This message is read during the **"read_queue_in"** activity of the Monitor. The following phase, **"generate_pending_goals"** concludes, by consulting the Self-Model, that some inputs are needed (***"goal_symbolic_state"*** and ***"initial_symbolic_state"*** which means the descriptions of initial and final object positions). Since such inputs are missing, it is necessary to look for, in the Self and Acquaintances Knowledge, who is able to inform about them. The input ***"goal_symbolic_state"*** is a result of ***"task_identification"*** capability of the HLP itself. Then, a new goal request is generated:

required_goal(task_identification,[hlp],400,common)

On the other hand, ***"initial_symbolic_state"*** is a

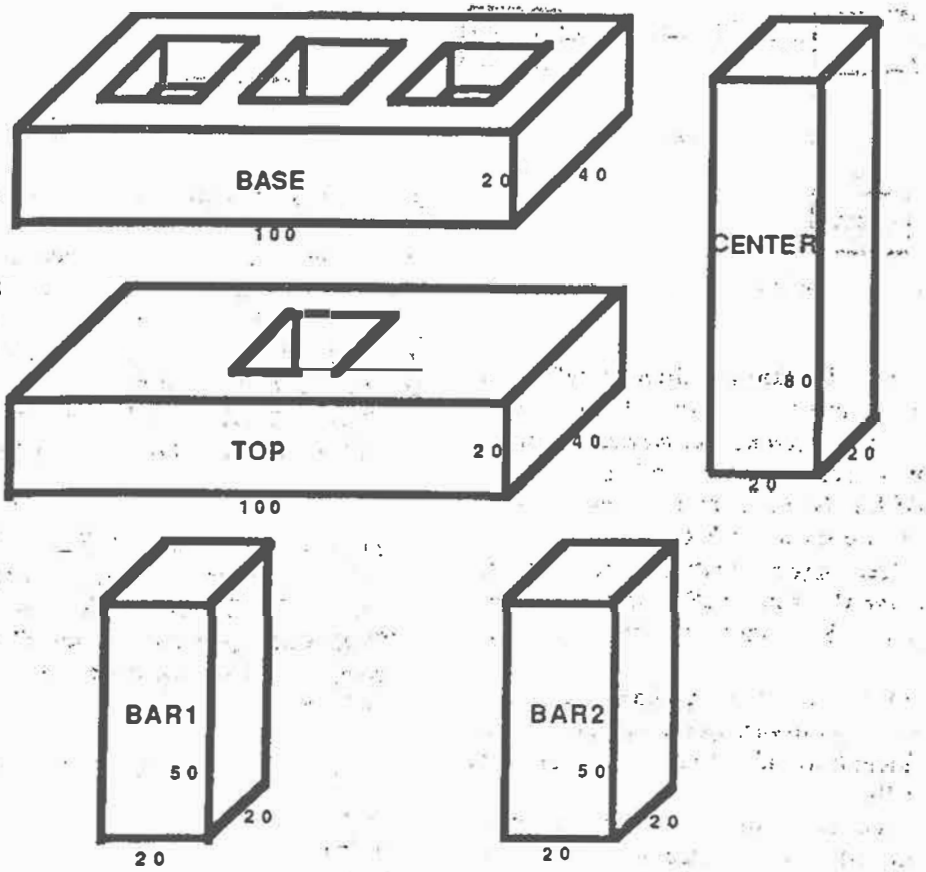


Fig. 5. Separated parts to be assembled.

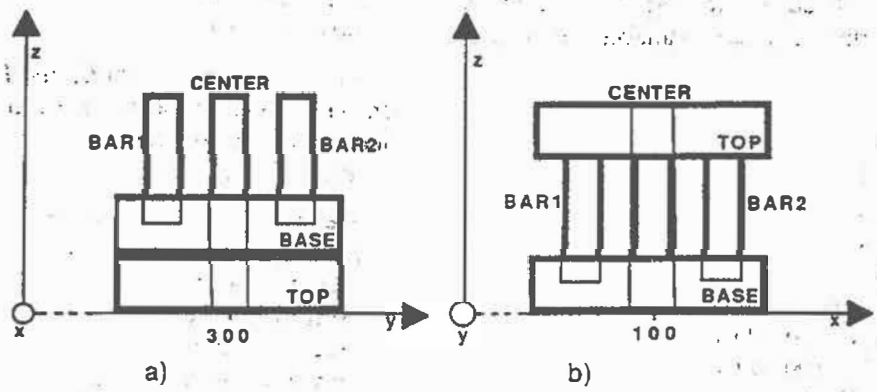


Fig. 6. Two possible assemblies of distinct parts.

result of ***"know_initial_state"*** goal of WM Intelligent System.

The only input for **task_identification"** is ***"task_description"***. The HLP Agent will ask the USER for this missing information.

At this moment the two goals of HLP are in the waiting state: **"task_identification"** waits for ***"task_description"*** and the goal **"generate_high_level_plan"** waits for ***"goal_symbolic_state"*** and ***"initial_symbolic_state"***. There are two pending goals in HLP Task Scheduler:

pending_goal(generate_high_level_plan, 1, [usr], waiting, 400)

pending_goal(task_identification, 2, [hlp], waiting, 500)

Figure 7 shows the state of the System, at this moment, in terms of goal requests.

The refreshing activity of WM agent allows the reception of:

required_goal(know_initial_state, [hlp], 400, common)

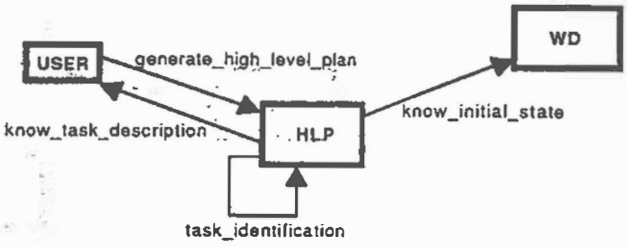


Fig. 7. Flow of cooperation among User, HLP and WD.

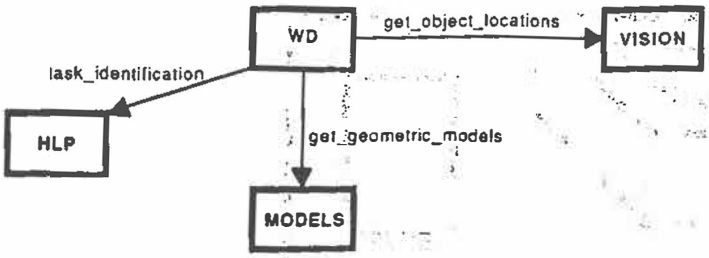


Fig. 8. Flow of cooperation among WD, HLP, Models and Vision.

For the execution of **"know_initial_state"** task, several inputs are missing (***"symbolic_features"***, ***"object_locations"*** and ***"geometric_models"***) which implies the need for cooperation between WD and HLP, VISION and MODELS. Needed interactions, in terms of WD goal requests, are shown in Figure 8.

VISION obtain **"get_object_locations"** request and needs ***"symbolic-features"*** (from HLP) and ***"geometric_models"*** (from MODELS). Figure 9 shows the new generated requests.

MODELS receives a **"get_geometric_models"** request, but once again ***"symbolic_features"*** are missing and HLP **"task_identification"** goal will again be invoked (see Figure 10).

Now let us pay attention to the overall system state. All agents are directly or indirectly waiting for **"task_identification"** goal execution. This goal needs a ***"task_description"*** input from the USER. **"Know_task_description"** is then fullfilled by the USER and the result is the following description made available to the HLP in-queue:

```
is_goal(on(base,floor))
is_goal(on(top,bar1))
is_goal(on(top,bar2))
is_goal(in(bar1,base))
is_goal(in(bar2,base))
is_goal(in(center,base))
is_goal(in(center,top))
goal_position(base,100,100,0,0)
goal_position(top,100,100,50,0)
goal_position(bar1,70,100,0,0)
goal_position(bar2,130,100,0,0)
goal_position(center,100,100,10,0)
```

The message ***"dk_available(task_description)"*** is sent to update HLP Self-Model. Now **"task_identification"** pending goal is changed to a ready state


Fig. 10. Flow of cooperation among HLP and Models.

and immediately executed because it has a higher priority than other activities. After the conclusion of this goal, the AAM of HLP is consulted to know who is interested in the achieved results. Therefore, and in accordance to the information about acquaintances WM, VISION, OBJECT MODELS, and LLP are informed about ***"symbolic_features"*** as follows:

```
symbolic_features([base, bar1, bar2, center, top])
```

Information about ***"dk_available(symbolic_features)"*** is sent to the previously mentioned agents.

The OBJECT MODELS now is able to provide VISION with ***"geometric_models"*** which enable this one to compute ***Object_locations"*** and send this information to WD and LLP.

WM is now able to execute the **"know_initial_state"** task giving the following results to HLP:

```
on(top,floor)
on(base,top)
in(bar1,base)
in(bar2,base)
in(center,base)
in(center,top)
```

Finally, since ***"dk_available(initial_symbolic_state)"*** has also been received, the ***"generate_high_level_plan"*** goal can be successfully executed giving to LLP the desired plan:

```
exec(put_on_floor(center))
exec(put_on_floor(bar2))
exec(put_on_floor(bar1))
exec(put_on(base,[floor]))
exec(insert(bar1,base))
exec(insert(bar2,base))
exec(put_on,(top[bar1,bar2]))
exec((insert(center,[base,top]))
```

**Example conclusions**

In the last section we have quickly shown how our system architecture is sufficient to enable simple

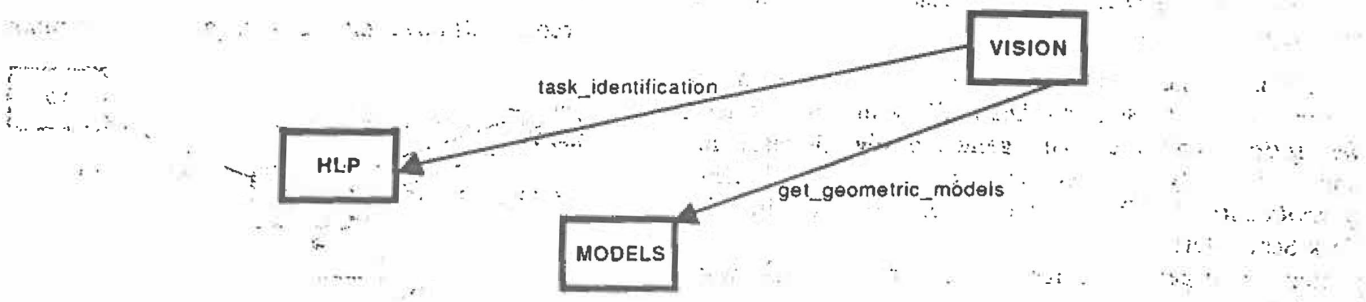


Fig. 9. Flow of cooperation among Vision, HLP and Models.

cooperative dialogues between different Agents in the Assembly Robotics framework. This conclusion does not imply that the use of more sophisticated mechanisms to support cooperation should not be used. Nevertheless, attention must be paid to the fact that more sophisticated mechanisms may degradate too much the overall performance of Intelligent Systems as regards time. In order to maintain a fair balance between communication and computation, our approach makes use of communication facilities only when really needed. The previous example shows this strategy.

One final word about implementation. The VISION agent is very time consuming and we are just dealing with 2D techniques. 3D characteristics are obtained through the interpretation of distance information, gathered from the scene by means of a grid of points projected by a Laser. Other agents, High-Level Planner, World and Objects Models and Low-Level Planner are already available. The more interesting one is HLP which uses some original ideas. HLP can be shortly described as non-linear and using an heuristic method of minimum cost to find the best sequence of non-conflicting actions. A pre-processing method is also used to delete goals already achieved. We would like to stress here that our HLP becomes very efficient when compared with traditional NOAH or STRIPS approaches. LLP, responsible for the translation of the high-level plan to the robot command language is also capable of inserting in the final procedures monitoring actions. LLP becomes ultimately responsible for the accomplishment of the orders given to the robot arm for execution.

## 9. CONCLUSIONS

We are trying to implement a Distributed Architecture including several Agents for Assembly Robotics tasks using Distributed Artificial Intelligence paradigm.

We already have identified specific and crucial features to be embedded in our Agents, as well as some peculiarities of their Reasoning process, Acquaintances and Communications mechanisms. Such Agents are now being developed in respect of their own competence and specific needs for cooperation.

A simple example illustrating how Agents are integrated in a cooperative community to solve a task was also described.


## ACKNOWLEDGEMENTS

We would like to acknowledge our Esprit Project 2256 partners (Krupp AE, Queen Mary WC, Framentec, NTU Athens, ISPRA, EDRC, Univ. Amsterdam, Volmac, CERN, IRIDIA, Iberduero, Labein, Ambar). All of them are contributing to a general architecture for cooperative systems. Our group is only working in a testbed based on assembly robotics. Special thanks are still due to T. Wittig (Krupp AE), A. Mamdani (QMW College), F. Arlabosse and E. Gaussens (Framentec) G. Stassinopoulos (NTU Athens), N. Avouris (JRC), who are also responsible for our interest in the subject. Nick Jennings was the first to write Archon Technical Notes about Self-knowledge and Acquaintances Models.


## References


1. M. Hunhs (ed.), *Distributed Artificial Intelligence* Research Notes in Artificial Intelligence Series (Morgan Kaufmann Publishers, Los Altos, California, 1987).
2. A.H. Bond, Les Gasser (eds.), *Readings in Distributed Artificial Intelligence* (Morgan Kaufman Publishers, Los Altos, California, 1988).
3. E. Oliveira, C. Ramos & R. Camacho, *AI* Systems cooperation for Robotics *Proceedings of First Workshop on Robotics and CIM, Lisbon* **section B** 9–19 (1989).
4. W. Walhster & A. Kobsa, "Dialogue-Based User Models" *Proceedings IEEE* **74,** 948–960 (July, 1986).
5. J. de Kleer, "An assumption-based TMS" *Artificial Intelligence* **28,** 27–162 (1986).
6. P.V.C. Hough, "Method and means for recognizing complex patterns" *U.S. Patent 3.069.654,* (1962).