Proc. ESPRIT Conf., Brussels, Belgium, 253-263.

COOPERATION IN INDUSTRIAL SYSTEMS

N.R.Jennings
Dept. Electronic Engineering
Queen Mary and Westfield College
Mile End Road
London E1 4NS
n_jennings@eurokom.ie

tel: +44-71-975-5358
fax: +44-81-981-0259

Summary

ARCHON is an ongoing ESPRIT II project (P-2256) which is approximately half way through its five year duration. It is concerned with defining and applying techniques from the area of Distributed Artificial Intelligence to the development of real-size industrial applications. Such techniques enable multiple problem solvers (e.g. expert systems, databases and conventional numerical software systems) to communicate and cooperate with each other to improve both their individual problem solving behavior and the behavior of the community as a whole. This paper outlines the niche of ARCHON in the Distributed AI world and provides an overview of the philosophy and architecture of our approach the essence of which is to be both general (applicable to the domain of industrial process control) and powerful enough to handle real-world problems.

1. INTRODUCTION

After more than a decade of successful exploitation there are now over 100,000 expert systems being used in hundreds of companies all over the world to solve complex problems in numerous domains [10]. Such systems have been particularly important and successful in the domain of industrial process control where conventional software and teams of operators were unable to cope with the demands presented by rapidly changing, complex environments [13]. However as expert systems technology has proliferated and individual systems have increased in size and complexity, new problems and limitations have been noted [20], [24]:

- **Scaleability**: the complexity of an expert system may rise faster than the complexity of the domain.
- **Versatility**: a complex application may require the combination of multiple problem solving paradigms.
- **Reusability**: several applications may have requirements for similar expertise. In a conventional system this has to be coded afresh in each new situation.
- **Brittleness**: expert systems typically have a very narrow range of expertise and are generally very poor at identifying problems which fall outside their scope.
- **Inconsistency:** As knowledge bases increase in size, it becomes correspondingly more difficult to ensure that the knowledge they embody remains consistent and valid.

One approach designed to circumvent these shortcomings and satisfy the ever increasing demands for speed, reliability and integration is to compartmentalize the problem solving into smaller, more manageable components and allow them to communicate and cooperate with each other (i.e. Distributed AI [2], [12] [14]). In such systems knowledge, resources, control and authority are distributed amongst community members who then work together, in a coordinated and coherent manner, to solve one or several problems.

The ARCHON[1] (ARchitecture for Cooperative Heterogeneous ON-line systems) project [16], [22] is concerned with the development of a framework which enables multiple problem solvers (some of which may be pre-existing) to be inter-connected so that they can communicate and cooperate with each other whilst solving complex, real-world problems. The real-world problems being tackled in the project are in the field of industrial systems: electricity management, cement factory control, robotics and control of a particle accelerator.

Within the field of Distributed AI many experimental platforms have already

1. The Archontes were the chief magistrates in Athens around 700BC. At that time there were nine, but with not very clear cut responsibilities and duties. These Archontes can be said to have formed a loosely coupled cooperative system with fuzzy boundaries for controlling the state.

been built and these broadly fall into two categories:

- systems which test a specific type of problem solver, a specific coordination technique or a particular domain

  eg ETHER [18], ECO [9], DVMT [19], ATC [4]

- systems which aim to be "general" to some degree

  eg MACE [11], MICE [7], CooperA [1]

However the weakness of these "general" systems is that they were not intended to be used in real-size problems (i.e. they lack the necessary power to solve large problems). The specific systems however may have the necessary power in a very narrow domain, but they are not generalizable. One of the major objectives of the ARCHON project is, therefore, to bridge this gap - to construct a cooperation framework which is both general enough and powerful enough to be used in a wide range of real-world industrial applications.

The remainder of this paper describes a typical ARCHON application in the field of industrial process control (electricity management) and highlights the characteristics of this domain which serve as important design forces. Section three gives an overview of the ARCHON architecture detailing the functionality of the various components and relating them to the problems identified in the previous section. Finally section four describes the underlying philosophy of our approach which leads to a system which is both general and powerful.

2. INDUSTRIAL PROCESS CONTROL

2.1 An Exemplar Problem

The two main applications within the project are concerned with electricity management and so a suitably simplified scenario (due to space limitations) will be used to illustrate the opportunities and benefits for cooperative interaction between problem solvers in this domain.

The interaction takes place between three problems solvers, each of which is an expert system in its own right (see figure 1). The low voltage diagnosis expert system is capable of detecting and diagnosing faults based on information (network status messages and customer telephone calls) about the low voltage network which it receives. There is also an expert system which carries out the same role in the high voltage network. The third expert system is capable of receiving information about the weather from various sensors around the country and of predicting the weather in the near future or of recalling (estimating) the weather conditions in a certain area at a certain time.

These three systems were developed at different times, by different groups of people and employ different problem solving techniques. However they all share some common ground which means there is potential for them to interact. The link between the high and low voltage systems is network connectivity: the low voltage

network being fed exclusively from the higher voltage network. So if there is no voltage at the higher level then there will also be none at the lower level, meaning there is no point in the low voltage system searching for a fault. The weather monitoring system is able to provide useful information for the diagnosing process because one of the main causes of faults (on both networks) is damaged power lines and the major cause of such damage is bad weather (either lightning storms striking plant equipment or strong winds causing equipment to be blown over). If the weather monitor can provide information regarding bad weather then this may serve as a focus for the problem solving process or be used to increase the certainty of hypotheses of equipment near a major storm.

At present the information generated by the three systems is merely presented to the operator who has to collate it and draw appropriate inferences, a very demanding task during an emergency. This burden is increased substantially if the individual sub-systems produce diagnoses which are inconsistent. Therefore it is desirable that the three systems work together, sharing information and results directly so that the operator is freed from the drudgery of collating reports and resolving conflicts and can concentrate on more cognitive, strategic-level tasks.

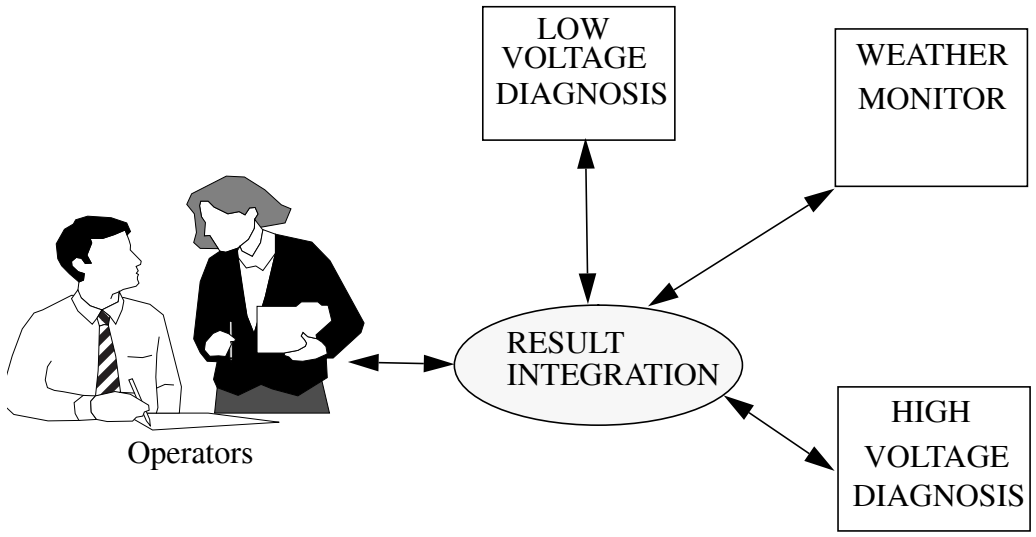

FIGURE 1 - Cooperative problem solving in electricity management

The objective of ARCHON is to provide a framework into which these systems may be integrated, enabling the potential for cooperative problem solving to be realised. In the above example, the main benefits include decreased operator workload, better utilization of problem solving resource (eg low voltage diagnosis system will not waste time trying to diagnose a fault which has been caused at the higher level) and increased confidence in hypotheses (since results have been cross-checked from several sources).

2.2 Major Design Forces

When targeting a system at a particular domain it is important to identify those characteristics which are likely to have greatest impact upon the design. After a

careful study of industrial systems the following major design forces were identified:

- There are a large number of existing software systems, each capable of solving some aspect of the overall problem using a paradigm which is most appropriate for that activity.
- Typically one (or more) operator(s) has to combine the possibly inconsistent results of several sub-systems in order to make informed decisions about the process as a whole.
- Typical applications are composed of many different types of generic function which have been automated in many different ways [13]:
  - *Diagnosis*: delivering an understanding of a world state given some information about this world
  - *Planning*: sequence a set of possible actions
  - *Control*: particular case of planning where actions are executable and low-level
  - *Supervision*: reflecting decision link between diagnosis of a dynamic system and the alternative actions needed for exceptional situation handling

Having identified these design forces it is important to analyse their effects. Firstly, the majority of Distributed AI systems make little or no attempt to integrate pre-existing systems. However in a domain in which there has been substantial investment in automation (such as industrial systems) it is clearly desirable from an economic viewpoint to be able to interconnect such systems. Such re-use has the additional benefits that software productivity can be substantially improved [3] and dependability can be increased (due to greater and more diverse usage). It is unlikely that such integration will be possible without *any* modification to the sub-system, but we aim to minimize the changes necessary. Pre-existence also implies that sub-systems are likely to be fairly sophisticated in nature (cf nodes of a neural network and Blackboard Knowledge Sources [8], [15]) and will be capable of a significant problem solving in their own right (they represent a *cohesive* body of knowledge and problem solving capability).

The desire to incorporate pre-existing systems combined with the nature and complexity of the domain means that problems of heterogeneity can occur at many different levels [21]:

- **Hardware Platform**: eg SUNs, Symbolics, VAXs
- **Operating System**: eg UNIX, VMS, MS-DOS
- **Programming Language**: eg LISP, Prolog, C, KEE, ART
- **System Type**: eg classical control, databases, expert systems
- **Architecture**: Expert Systems - Frames, Blackboards
- **Semantics**: Understanding and distribution of concepts
- **Purpose**: Distribution of capabilities

Typically Distributed AI work has made simplifying assumptions about the uniformity of problem solvers or their domain of application, but as the above classification highlights heterogeneity is a pervasive phenomenon in real world problems. We argue that the form of heterogeneity which has the most significant impact upon the design of a cooperation framework concerns semantics and purpose [21], and propose that sophisticated agent and information modelling facilities (see section 3) are required to cope with these problems.

Many of the typical advantages of distributing both data and control when problem solving (as expressed in [2], [12] [14]) also apply to ARCHON applications; however the predominant benefits in this type of environment include:

- Enhanced problem solving

  by sharing knowledge and data, systems may be able to make use of information which would not have been available normally (due to resource limitations, for example) and thus problem solving may be faster, of a higher quality, more accurate and so on.

- Ease the burden on the operator

  in emergency situations operators are typically faced with a barrage of possibly contradictory information from multiple sources which they have to collate, interpret and act upon. By allowing the sub-systems to communicate with each other directly the operator's cognitive load is reduced considerably and he can concentrate on strategic level considerations.

- Increased reliability

  because problem solving capability may be available at multiple sites within the community, failure at one node does not lead to total system collapse (i.e. system exhibits graceful degradation of performance).

3. AGENT STRUCTURE

The aim of the ARCHON framework is to create an environment in which social interaction is possible. However the problem solvers of figure 1 have no cooperative knowledge, they were conceived and built as stand alone systems and therefore do not know how to behave or participate in an environment which contains other entities. All they posses is the domain-level knowledge necessary to solve domain-level problems, e.g. how to diagnose faults in low and high voltage networks and how to predict the weather - we call such systems **Intelligent Systems**. In order for cooperative behaviour to be realised, the intelligent systems must be augmented with knowledge which enables them to engage in social activities (eg to initiate, maintain and respond to cooperative situations, to be able to asses the needs of the community as well as their role within the community and so on). Within ARCHON, this social awareness is achieved by enhancing each intelligent system with a series of modules embodying the necessary social knowledge[2]. This collection of modules is collectively referred to as the **ARCHON layer** and the combination of an intelligent

system and its ARCHON layer as an **agent**.

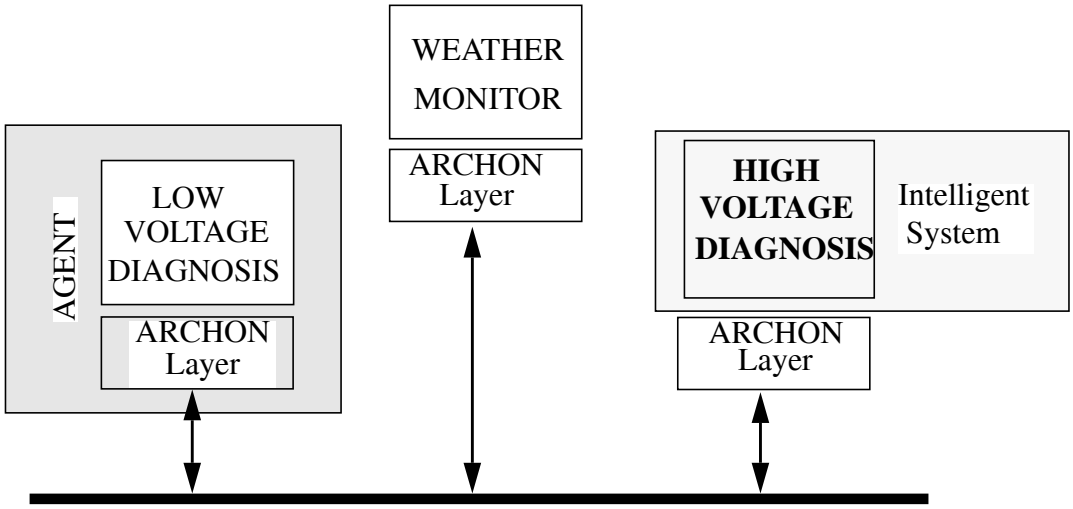

Figure 2 - An ARCHON community

The movement to a cooperative environment means a fundamental change in the requirements of an individual intelligent system, which the ARCHON layer must support. Whilst in the asocial situation depicted in figure 1, each system plans its activity exclusively on the basis of domain knowledge; in a social context an agent's activities are planned on the basis of both domain knowledge and social knowledge. An ARCHON agent, therefore, has two distinct (but related) roles to satisfy: that of a team member acting in a community of cooperating agents and that of an individual. Much of the early Distributed AI work concentrated almost exclusively on the former and paid scant regard to the latter; however ARCHON, and contemporary DAI in general [5], [6], places greater emphasis on the role of the individual. Therefore when designing the ARCHON layer both aspects should be catered for:

- Control (direct) local problem solving component

  eg which tasks to launch, when they should be launched, their relative priorities and how best to interleave their execution, recovery from local exceptions

- Coordinate local activity with that of others within the community component.

  eg when and how to initiate cooperative activity, which cooperation protocol (client-server, contract net, etc.) to employ, how to respond to cooperative initiations, which activities require synchronization.

These two perspectives provide a design rationale and separation of concerns upon which the ARCHON layer architecture can be based - see Figure 3. The monitor is responsible for controlling the local problem solving activity while the planning

2. Using this approach means that the cooperative know-how is also conceived and implemented in a distributed manner. The alternative, one meta-system controlling all the others, was rejected because such a controller would become a computational and communication bottleneck and also because the complexity of the problem being tackled would prevent such a system from maintaining a consistent and complete view of the entire problem solving process (i.e. it would posses bounded rationality [23]).

and coordination module (PCM) is responsible for controlling an agent's cooperative interactions. There is, however, a grey area between these two modules which has to deal with the impact of local decisions on the global perspective and of global decisions on the local activity - as choices about such interactions effect both the monitor and the PCM they are dealt with and agreed by both modules.

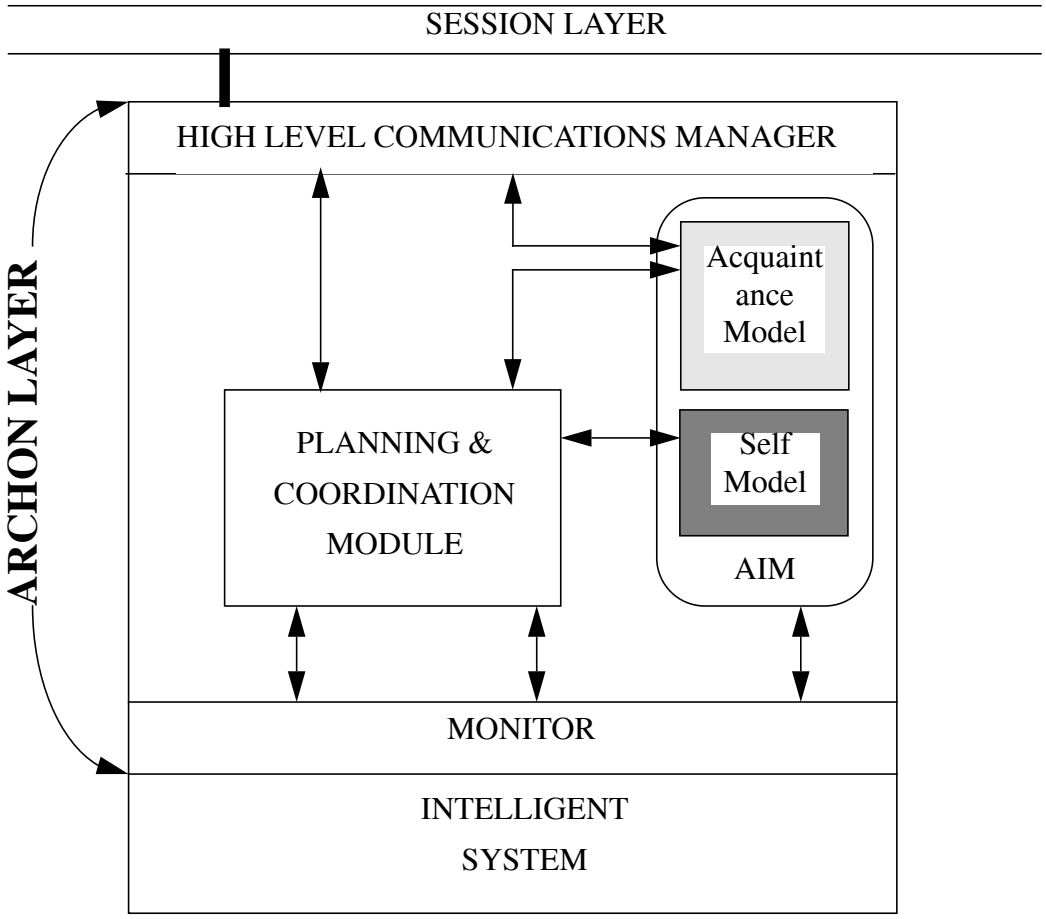

Figure 3: Detailed structure of the ARCHON layer

The other components of the ARCHON layer are there to support these two modules. The High Level Communication Module (HLCM) supports the types of dialogue necessary for decentralized problem solving and coordination; offering facilities such as message scheduling, message filtering and intelligent addressing. The HLCM interfaces to an extended Session Layer[3] which means ARCHON communities can be installed over networks conforming to the OSI standard. AIM (the Agent Information Management module) provides an object-orientated information model, a query and update language to define and manipulate the information and a distributed information access mechanism to support the remote access and sharing of information among agents.

3. The session layer has been extended to provide facilities appropriate for cooperative problem solving. For example, the standard does not cater for point to multi-point connections, something which is essential if broadcasts to groups of agents are required.

It is well acknowledged in many fields (including sociology, economics, politics and Distributed AI) that sophisticated cooperation requires participants to have some knowledge of other team members. The acquaintance models [2], [11], [21] provide exactly this knowledge - they are an abstract description of the other agents with which the modelling agent has to interact and are used by the PCM and the monitor for determining and mediating cooperative interactions. The type of information they maintain include: the capabilities of others, their intentions and plans, current state of processing, what information they can generate/are interested in and so on [21], [22]. The self model provides a meta-level description of the underlying intelligent system which the monitor can reason about when controlling the local system - the information to be maintained includes the status of active tasks, which tasks are pending, which tasks are waiting for which pieces of information, the relative priority of the various tasks, for example.

4. ACHIEVING GENERALITY AND POWER: A HYBRID APPROACH

By aiming to produce a cooperation framework which has a degree of generality, rather than constructing specialized systems for each particular application, it is important that the approach adopted is flexible yet still expressive enough. The two main weapons with which we attack this problem are both outlined in this section:

- Combining pre-compiled behaviours and reflective planning components to guide agent activities.
- Using domain independent structures with domain dependent instantiations

One way of increasing the power of a system is to compile reasoning and action [17] - such "chunks" are usually referred to as behaviours. However systems in which all action is pre-compiled (reactive systems) have certain inherent limitations, the most pertinent of which is the fact that behaviours are usually specific to a certain problem and application. In contrast, systems which perform reflective reasoning and planning at run-time are applicable to a wider range of problems (particularly if this reasoning is based on domain independent structures - see below), but are typically more expensive in terms of the resources they consume. The ideal approach therefore, is to take the power offered by behaviours and combine it with reflective reasoning (generality) of the appropriate level. The characteristics of these two problem solving paradigms can then be used in the parts of the architecture to which they are best suited:

- Monitor: Requires fast response to large number of domain specific events and is difficult to generalise

  ⇨Predominantly behaviour based

- PCM: Small number of social activities, need to reflect the particular run-time situation and offers greater scope for general descriptions.

  ⇨Predominantly reflective

The second method is to make use of domain independent structures. Within

the ARCHON architecture such structures include the communication facilities, the agent modelling facilities and the information modelling facilities (AIM). However due to space limitations the discussion will focus on the agent modelling facilities. The idea behind the acquaintance models is that they should embody most (ideally all) of the information necessary for supporting social activity and for evaluating the status of the community from the modelling agent's perspective. Similarly the self models should embody the information necessary for controlling the underlying intelligent system and for assessing the agent's local situation. The semantics of these models are then given by a more or less generic control mechanism[4]. Hence generality is achieved because substantial parts of control mechanism can be common to all agents, but power is retained because domain complexity is represented in the models.

To illustrate this point we will return to the electricity management problem specified in section 2.1. The weather monitor's model of the low voltage diagnosis agent will represent the fact that it is interested in lightning strikes when trying to diagnose faults:

Modelling Agent: Weather Monitor
Modelled Agent: Low Voltage Diagnosis Agent
Slot Name: Interests
Slot Structure: {<$Name_1$, $Cond_1$>,..... <$Name_n$, $Cond_n$>} **(TEMPLATE)**
Slot Structure: {<LIGHTNING-STRIKES, **(INSTANTIATION)**
WORKING-ON(DIAGNOSE-FAULT)>}

The semantics of this part of the agent model can be provided by the following generic control rule[5]:

IF knows (agent(?A), information(?I)) **AND**
($\exists$ <$Name_x$, $Cond_x$> $\in$ INTERESTS(agent(?A), acquaintance(?A2)):
equal($Name_x$, information(?I) **AND**
is-true($Cond_x$))
THEN can-help(agent(?A),
acquaintance(?A2),
send(agent(?A), acquaintance(?A2), information(?I)))

5. CONCLUSIONS

This paper serves as a mid-term progress report for the work currently being undertaken within the ARCHON project. It highlights those features of the project which distinguish it from previous work in the field of Distributed AI and hence defines the project's niche. The boundary of the range of problems being tackled has been described and an approach which leads to both a powerful and general

4. The underlying premise is that meaningful behaviour can be defined by the **structure** of the models and the actual contents merely provide the context for interpretation.

5. The meaning of the predicates is self explanatory. can-help(a1, a2, X) means that agent a1 is in a position to help acquaintance a2 by doing X. Whether it actually decides to do X will be based upon the current situations of a1, a2 and the rest of the community.

framework has been outlined. The ARCHON architecture, as it exists at present, and the major design forces which give rise to this architecture have also been described.

At this stage in the project we are engaged on several simultaneous activities: specifying in greater detail the exact separation of concerns between the various components of the ARCHON layer, integrating (at the software level) the various components of the architecture, reconceptualising applications to ensure maximum utility is made of the cooperating systems metaphor and also verifying our architecture against the real world scenarios which exist within the project.

ACKNOWLEDGEMENTS

This paper represents the work of the whole ARCHON consortium which consists of the following partners: Krupp Atlas Elektronik, JRC Ispra, Framentec, QMW, IRIDIA, Iberduero, Labein, Electricity Research and Development Centre, Amber, Technical University of Athens, FWI University Amsterdam, Volmac, CERN and University of Porto.

In particular the following have contributed significantly to the current state of the ARCHON architecture: Abe Mamdani (QMW), Thies Wittig (Krupp) and Erick Gaussens (Framentec) to the overall architecture design; Claudia Roda (QMW) to the HLCM; Jochen Ehlers (Krupp) to the PCM; George Stassinopoulos, T Tsatsaros and M. Spyropoulou (all Univ Athens) to the Session Layer Work; Francois Arlabosse, Daniel Gureghian and Jean Marc Loingtier (all Framentec) to work on the Monitor and Behaviours; Frank Tuijnman and Hamideh Afsarmanesh (both Univ. of Amsterdam) to work on AIM and Eugenio Oliveira and Long Qiegang (Univ Porto) to work on generic rules.

REFERENCES

[1] AVOURIS,N.M., LIEDEKEKERKE,M.H.V. & SOMMARUGA,L. (1989). Evaluating the CooperA Experiment. Proc. of 9th Workshop on Distributed Artificial Intelligence, Seattle.

[2] BOND,A.H. & GASSER,L. (1988). Readings in Distributed Artificial Intelligence. Morgan Kaufmann.

[3] BROOKS,F.P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. Computer 20 (4) pp 10-19.

[4] CAMMARATA,S, McARTHUR,D. & STEEB,R. (1983). Strategies of Cooperation in Distributed Problem Solving. Proc IJCAI 1983, pp 767-770.

[5] DEMAZEAU,Y & MULLER,J.P. (1990). Decentralized AI. Elesevier Science Publishers.

[6] DEMAZEAU,Y & MULLER,J.P. (1990). Decentralized AI Vol II. Elesevier Science Publishers.

[7] DURFEE,E.H. & MONTGOMERY,T.A. (1989). MICE: A Flexible Testbed for Intelligent Coordination Experiments. Proc. of 9th Workshop on Distributed Artificial Intelligence, Seattle

[8] ENGELMORE,R. & MORGAN,T. (1988). Blackboard Systems. Addision Wesley.

[9] FERBER,J. (1989). Eco-Problem Solving: How to Solve a Problem by Interactions. Proc. of 9th Workshop on Distributed Artificial Intelligence, Seattle

[10] FEIGENBAUM,E.A., McCORDUCK,P. & NII,H.P. (1988). The Rise of the Expert Company. Times Books.

[11] GASSER,L., BRAGANZA,C. & HERMAN,N. (1988). Implementing Distributed AI Systems Using MACE. in [14].

[12] GASSER,L. & HUHNS.M.N. (1990). Distributed Artificial Intelligence Vol II. Pitman.

[13] GAUSSENS,E. (1990). Needs and Opportunities for Expert Systems in the Process Control Field. Vacation School for Process Control, University of Strathclyde, Scotland.

[14] HUHNS.M.N. (1989). Distributed Artificial Intelligence. Pitman.

[15] JAGANNATHAN,V., DODHIAWALA,R & BAUM,L.S. (1989). Blackboard Architectures and Applications. Academic Press.

[16] JENNINGS,N.R. (1991). An Architecture for Cooperating Systems. Artificial Intelligence and Simulation of Behaviour Quarterly, Special Issue on Distributed AI, 76.

[17] KISS,G. (1991). Layered Architectures for Intelligent Agents. IEE Colloquium on Intelligent Agents.

[18] KORNFIELD,W.A. & HEWITT,C.E. (1981). The Scientific Community Metaphor. IEEE Trans. on SMC. 11, 1, pp 24-33

[19] LESSER,V.R. & CORKILL,D. (1983). The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. AI Magazine, pp15-33.

[20] PARTRIDGE, D. (1987). The Scope and limitations of First Generation Expert Systems. Future Generations Computer Systems, 3, 1, pp 1-10.

[21] RODA,C., JENNINGS,N.R. & MAMDANI, E.H. (1991). The Impact of Heterogeneity on Cooperating Agents. Proceedings AAAI Workshop on Cooperation among Heterogeneous Intelligent Systems, Anaheim, Los Angeles.

[22] RODA,C., JENNINGS,N.R., & MAMDANI,E.H. (1990). ARCHON: A Cooperation Framework for Industrial Process Control. in Cooperating Knowledge Based Systems 1990 (ed DEEN,S.M.), pp 95-112, Springer Verlag.

[23] SIMON,H.A. (1957). Models of Man. Wiley.

[24] STEELS,L. (1985). Second Generation Expert Systems. Future Generations Computer Systems. 1, 4, pp 213-221