scope: *diagnosis*
contribution: *application*

ARCHON: framework for intelligent co-operation

by T. Wittig, N. R. Jennings and E. H. Mamdani

The paper describes the work that has taken place in the ARCHON Project, ESPRIT project P-2256. The consortium has developed a general-purpose architecture, which can be used to facilitate co-operative problem-solving in industrial applications. The paper describes the need for a multiple agent approach for industrial applications, outlines the benefits which can be accrued by adopting this paradigm, and describes the key difficulties which must be faced when building a multi-agent system in this domain. Details of the ARCHON architecture are presented, including a description of the main functional components and their realisation in a hybrid agent model. An example of co-operative fault diagnosis in an electricity management application is described in order to provide a clear illustration of the working of the ARCHON architecture, and to provide a concrete example of the potential benefits of a multi-agent approach.

1 Application challenge

The initial objective of the ARCHON Project was to build a software architecture that would allow pre-existing expert systems, dealing with different aspects of decision-making of a given complex environment or a system, to co-operate in a mutually beneficial way. Given that all the expert systems that engage in such a co-operation are in fact responding to the same overall environment, any properly designed co-operation would thus increase the effectiveness of each one. From such an objective it soon becomes apparent that the co-operation cannot be restricted only to expert systems, but the system also needs to include databases and other computational systems. ARCHON therefore uses the term 'intelligent systems' to refer to the domain-dependent systems to be incorporated within the co-operative architecture. Furthermore, while considering co-operation, we also need to attend to other forms of co-ordination that help to increase the effectiveness of a generic system such as ARCHON.

The main design feature of ARCHON is therefore that it places emphasis on a loose coupling as a means to increase co-operation between a set of computational systems. Thus, the computational systems that participate in mutual co-operation are conceived of as almost autonomous and capable of completing their allocated tasks without much reliance on other systems within the community, but benefiting from each other's activities through the co-operation mechanisms. ARCHON provides various modes of co-operation including passing of unsolicited information, one use of which is that computational systems can anticipate future needs from other co-operating systems; this mode of co-operation is sometimes referred to as 'pro-active'.

Just as the ARCHON approach is not restricted to co-operation among expert systems, it is also not restricted to pre-existing computational system. ARCHON can be construed as providing a means for the creation of a 'federated' computational environment to support an organisation's computational needs. ARCHON is more than a transient technology for integrating legacy systems. Its design objectives were always for the inter-working of semi-autonomous agents. It can complement integration architectures that provide for tight coupling of systems, such as client/server architectures in which a client would demand a service and a server is mandated to provide that service. ARCHON agents may enter into a client-server relationship with each other for a contracted set of tasks, but are never designed (pre-destined) to perform one or other of those roles. ARCHON agents can pass unsolicited information to their acquaintances, leaving it to the recipient to decide what to do with it (e.g. whether and when to enter a form of eager evaluation in anticipation of future requests from the sender).

Having said this, it must be pointed out that, within

the ARCHON Project, there were several application evaluations, all of which had at least some pre-existing systems. Furthermore, the Project itself did not evaluate how the ARCHON approach might complement an existing client-server type integration environment.

1.1 ARCHON applications

When deciding on the application areas to which the ARCHON system should be applied first, the following points were considered:

- the application should have some 'intelligent systems' already running.
- it should consist of subsystems that require co-ordination (usually by an operator), and thus can be turned into 'agents'.
- it should be of sufficient complexity to provide the challenge for the research but at the same time be manageable within this type of project.

Based on these criteria, it was decided to concentrate on industrial supervision and control applications because

□ industrial supervision and control applications are usually built in a modular fashion, allowing separation of subsystems more easily as, for example, applications in the business domain.
□ expert systems, at least as prototypes, could be found more often in this domain.
□ most partners were working in that domain, either as a software company, system house, research institution or user.

In the end, the decision was made for *electricity management*. Although this may not be the most obvious application for distributed AI (DAI), it has a number of advantages. The most prominent ones are

- the application problem as seen from a DAI point of view is 'balanced', i.e. it is not too simple but also not too complex. A number of subsystems exist in such applications, each with a specific task but sometimes with overlapping capabilities. In the two specific applications we chose, a number of knowledge-based systems had already been developed or were under development at that time. This meant that we could really concentrate on the important part of co-ordination among these systems.
- the application domain was familiar to some of the research partners who had previously developed expert systems in this area [1, 2].

There is, of course, a danger in focusing entirely on one application domain. In order to ensure generality of the architecture, we also included so-called 'application studies'. These are smaller exercises in applying the emerging concepts to quite different applications. The ones we were looking at are

□ *robotics*; application of the ARCHON framework in the area of intelligent robotics with the objective to implement a flexible production cell. The subfields for co-operation are robot control, sensing (proximity sensor in the robot arm), planning for task assistance, and vision/scene analysis [3]. A second, closely related application study was carried out for a robot arm with six degrees-of-freedom in a laboratory environment. The arm is equipped with a variety of sensors, including a vision system and force sensing. The control hardware consists of a VME crate containing a variety of processors, connected through a LAN to a SUN workstation. This system, in particular, will allow testing of the ARCHON software in a real-time environment, which is relevant to later process control applications.
□ *cement production control*; the purpose of this application study is to investigate the co-ordination, through co-operation, of the various control systems of a cement production process [4, 5]. Individual controllers assisted by expert systems already exist and are currently tied together by a hierarchical and tightly coupled system. Within ARCHON, this rigid organisation has been replaced by a number of semi-autonomous loosely coupled agents; the precalciner agent, the kiln burner agent and the clinker cooler agent, each controlling a number of tasks that have to be co-ordinated.
□ *fault finding and operational help in the control system for particle accelerators*; particle accelerators have been developed to create high-energy beams of particles to investigate the subatomic world and the fundamental nature of all matter. The CERN proton synchrotron (PS) complex comprises particle accelerators of different types, accumulation and storage rings, and beam transfer lines, controlled through a large computer network of about 20 minicomputers and 150 microcomputers, interfaced by a data acquisition system to the components of the accelerators (e.g. the power supplies for bending magnets, the radio frequency cavities for acceleration and the complex beam measurement devices).

Running an accelerator is like controlling a large industrial process. Fault finding and repair in the PS control system have become time-consuming and difficult, and require a significant amount of staff resources [6]. The systems to be integrated are a large relational database, a control system consisting of a large computer network and several separate expert systems. The domains are beam diagnosis and control, control system diagnosis, and alarms treatment. The task of these expert systems is to find faults in the control system to help maintenance and control of the accelerator. Most of the static knowledge for the expert systems has to be read from the database, and dynamic information has to be read on-line from the control system to be diagnosed.

2 DAI use for industrial applications

Large systems are built in a distributed fashion in order to master complexity. Ideally, this should mean a separation of control and execution to make the control part more explicit. Consequently, the next level to explore is system design. Complexity of systems has two different aspects; it refers to the solution of the primary application problems, which determines the functional requirements of a system and, on the other hand, it refers to issues such as security, maintainability, flexibility and adaptability to changing requirements during the lifecycle of the systems. These are issues that establish the non-functional requirements of a system.

Essentially, these non-functional issues are those addressed by the co-operative systems approaches like ARCHON. It means dividing a system into several smaller and dedicated systems, whereby not only the execution becomes distributed but, most importantly, the control becomes decentralised. Obviously, the control part of each of these dedicated systems has to deal with the solution of its own problem domain. In addition, it has to control the co-ordination with the other systems, a task that is dealt with centrally in non-distributed monolithic systems. By distributing this control and assigning it to the individual systems, allowing them to control not only themselves but also the way they interact, the control complexity is reduced.

2.1 *Benefits of DAI in industrial applications*

The general benefits associated with using a DAI approach stem from the distribution of control and data and the increased software modularisation that can be achieved [7–9]:

- modularity; traditional advantages from software engineering, including decreased component complexity and greater clarity.
- speed; sub-systems can operate in parallel.
- reliability; system can continue even if part of it fails.
- knowledge acquisition: easier to find experts in narrow domain.
- reusability; small independent system could be part of many co-operating communities.

In addition to the above, within the domain of industrial systems, the benefits of enhanced problem-solving, easing the burden of the system operator and managing the data exchange among distributed systems, are especially pertinent [10, 11]. Individual problem-solving can be enhanced in quality as well as speed by sharing information and processing in an efficient manner between community members. If an operator is in charge of several systems working on the same process, the operator has to manually pass information between the systems and ensure their activities are co-ordinated. However, by automating this interchange, the operator can concentrate on the higher level (cognitive) actions for which the operator is better suited. Finally, reliability is enhanced because if part of the system fails, the performance should degrade in a graceful manner as both the control and data are distributed.

2.2 *Design constraints of industrial systems*

When considering a DAI approach to solve industrial problems, several important characteristics of the domain need to be considered:

- □ there is a substantial amount of pre-existing software.
- □ industrial systems are complex and require many diverse types of activity to be performed.
- □ the operator is an integral component of the problem-solving community.

2.2.1 Pre-existing software: typically within the domain of industrial systems, software has been developed in an *ad hoc* fashion when the company has perceived that certain functions could profitably be automated using available technology. The result is that such companies possess a large number of stand-alone systems, developed at different times, by different groups of people and utilising different problem-solving techniques. These systems all operate within the same physical environment (i.e. take input from and produce output to the same process) and could benefit from interaction (sharing of information and problem-solving expertise) with other such stand-alone systems. However, as they were not constructed with the aim of integration in mind, they employ techniques and representations that are best suited to the particular problem(s) for which they were designed; ensuring that data/knowledge formalisms and the associated reasoning techniques are specific to that particular implementation. Faced with this situation, there are several courses of action open to the system designer including

- rebuilding all the existing software in a common environment, so that they all share common representations, reasoning mechanisms and knowledge semantics.
- constructing a framework into which the existing systems can be incorporated (with minimal modifications*) and allow them to interact with each other.

In most cases, the latter option is preferable because of its lower cost, resource usage and lower general risk. However, this means that problems associated with heterogeneity become an important issue and require attention within the agent architecture. There are many different levels of heterogeneity that need to be dealt with, ranging from different operating systems and programming languages to issues associated with the inherent distribution of tasks within the community and the overlap in the understanding of domain concepts between agents [12].

2.2.2 Diverse activities: large industrial control applications are inherently heterogeneous with respect to the various subsystems involved. Subsystems are targeted to specific and quite diverse tasks [13]:

- □ diagnosis; deliver an understanding of world state given some information about this world.
- □ planning; sequence of a set of possible actions.
- □ control; particular case of planning where actions are executable and low level.
- □ supervision; reflecting a decision link between diagnosis of a dynamic system and the alternative actions needed for handling exceptional situations.

The wide range of problem-solving techniques necessary for each of these sub-areas means that a common technique and knowledge/data representation would be unfeasible, and hence issues related to heterogeneity are important even when constructing completely new systems. To obtain the necessary complexity, different

* Typical examples of the modifications that have to be made include allowing the co-operation framework to affect the control instance of the system and altering the system's presentation of information to the user (to incorporate the fact that the system is now a team member and not merely an individual).

techniques will be needed for different tasks and, even within a single generic task, more than one technique may be employed (e.g. in diagnosis there may be a heuristic-based component and a model-based component). For example, an alarm analysis system in an electricity distribution network has to cope with large numbers of incoming events and map these to the topology of the network held in huge real-time databases. In contrast to this, an electricity load management system that plans and controls the overall load of a network only deals with a small number of real-time events, but instead has to cope with non-monotonic planning based on crude qualitative models of the consumers in the network. Therefore, it is not surprising that these systems not only differ widely in their software structure, but also often run on quite different hardware platforms.

2.2.3 Operator involvement: due to the critical nature and potential risks associated with industrial systems, it is inevitable that human operators will remain an integral component for the foreseeable future [14]. Therefore, when designing a community of co-operating agents, it is essential to ensure the operator is included as an active problem-solving member, able to volunteer information, carry out problem-solving tasks, focus activity etc. Some of the issues that need to be addressed include the allocation of tasks between operators and the artificial agents, how to design features of the interaction so that the best capabilities of operator and computational agents are utilised, how authority can be implemented etc. These issues are discussed more thoroughly elsewhere [15, 16].

3 ARCHON functional architecture

3.1 Using the application context to derive requirements

In order to describe the required functionality of the ARCHON approach, we use the 'idealised operator analogy'. We assume a set of independent supervisory and control (S&C) systems, each controlled by an operator with the only co-ordination taking place among the operators (depicted on the left of Fig. 1). The operators share a particular overall goal, e.g. the economic efficiency of a power distribution network or the safe operation of a large vessel. Each operator derives their personal aim for controlling their S&C system from such an overall goal. Naturally, they are aware that they are not working in isolation; they are aware of the other operators and have some understanding of what their tasks are and what they can achieve. This knowledge enables them to either ask for help or to respond to requests from the other operators.

This operator level is the target of ARCHON, and the problem is to identify the functionality a system requires in order to act on this level. The right of Fig. 1 lists the main requirements for an architecture that, in the end, will achieve the co-operative link between independent systems. We briefly discuss these points and indicate how they are mapped into the functional architecture of ARCHON.

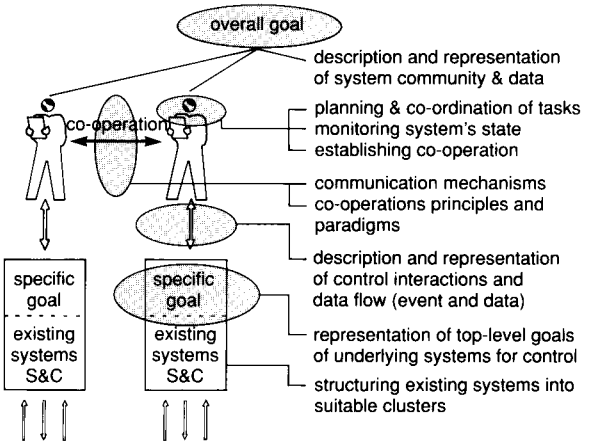

Fig. 1 Requirements for a co-operative framework

3.1.1 Overall goal: although the general goal of an application has to be represented somewhere in a suitable form, the heterogeneity of the application suggests a more loosely coupled approach. Consequently, we do not represent a 'common goal' anywhere in our architecture but only goals of the agents which, taken together, form the overall goal of the community. This is one of the most important tasks the designer of such a co-operative system has to consider, and it is very closely related to the way in which the community is represented within each agent. No agent has to model all the others, only those with which it is likely to interact. By representing skills, interests and goals of its acquaintances, an agent is able to specifically involve others in its own problem-solving objective and at the same time respect their autonomy [10, 11, 17].

3.1.2 View of their own system: just as an operator works on their S&C system, the ARCHON framework functionality has to provide this view. This involves monitoring and assessing the system's state on the one hand, and providing the means to interact with and control it on the other. To do this, the control interactions for the domain system have to be described and represented in a suitable way. In ARCHON, this is primarily done through an event-based mechanism that allows time and data to be captured. There are two different ways to respond to events from the domain-system; reactive and reflective. The first one relates to 'standard' responses in nominal situations, i.e. when the system or the application is behaving as expected. Reactive responses do not require explicit reasoning and can be dealt with by precompiled plans. In exceptional situations, however, such precompiled plans would fail. Much in the same way as an operator has to think about such problems, ARCHON has to provide means for explicit reasoning, i.e. it has to contain a reflective component.

Operators usually have a certain amount of freedom in assigning tasks to their S&C system, based on their knowledge of the system's tasks and goals. In order to flexibly control the system through ARCHON, its top-level goals and accessible tasks have to be represented. An ARCHON system designer again has to pay attention to this point; as long as the domain system does not permit different control options, there is not much point

in building a co-operative framework on top of it. It may well be that the designer has to cluster existing systems into suitable bundles to achieve this flexibility of control.

3.1.3 View on the community: based on the assessment of the state of the domain system, the need for help from other agents may arise. Equally, pre-empting urgent requests from other community members, an operator may volunteer certain information derived from their system to others. Through the representation of skills, interests and goals of other agents, coupled with the appropriate reasoning facilities, ARCHON provides this functionality. Being the central point of a co-operative framework on top of existing systems, this functionality should be as independent from any domain systems as possible. In fact, as described later, the current implementation provides a number of such independent rules for detecting the need for co-operation and for determining the best way of establishing it.

3.2 General architecture and its mapping to the requirements

We explain the architecture of ARCHON in two steps; first, the general modules that can be derived from the requirements identified in the previous section are presented, and secondly we show how these modules map with the 'operator analogy'.

Fig. 2 sketches the modules of the ARCHON layer and shows the interface to the intelligent (domain level) system. In this architecture, there are in fact two levels of 'intelligence'; on the domain level and, more important in this context, on the co-ordination and co-operation level. Each ARCHON layer is in itself a knowledge-based system, reasoning about its domain system and the co-ordination within the community, but not solving any of the domain problems.

The architecture needs a communication facility, the high-level communication module (HLCM). It is high-level as it not only provides the communication facilities

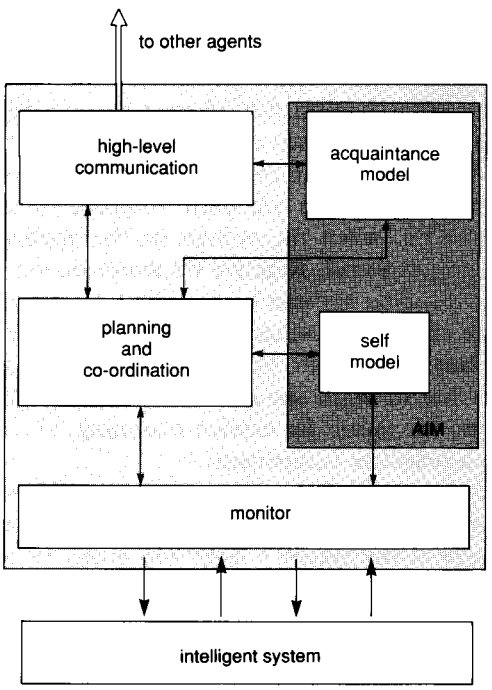

Fig. 2 Agent architecture

(achieved through a session layer implementation), but also services such as intelligent addressing and filtering. For example, if the domain system produces a result that may be relevant for other agents, the planning and co-ordination module (PCM) just asks the HLCM to send it to all interested agents without specifying them.

The agent information management (AIM) module provides an object-oriented information management model, a query and update language to define and manipulate the information, and a distributed/federated information access and retrieval mechanism to support the remote access and sharing of information among agents. AIM is used to store both the application data and the agent models [18], the latter are required for the agent to reflect about its role in the community.

The agent acquaintance models (AAM) contain representations of other agents in the community in terms of their skills, interests, current status of workload etc. [17]. Agents will not necessarily maintain models of all community members; it is more likely that each agent will merely model a subset of the total community. This subset is constructed on the basis of similar interests/capabilities or the ability to provide services that the local agent cannot perform (e.g. solve certain goals or furnish particular pieces of information).

Much like the AAM models other agents, the self model (SM) is an abstract representation of an agent's domain system. It primarily contains information about the current state of this system, i.e. its workload, or what tasks are being executed, but it also embodies the precompiled plans (behaviours). These plans are accessed by the monitor, which is responsible for the control of the intelligent system and for the passing of information to and from it. As well as representing information about different entities, there is a clear distinction in the way in which the self and acquaintance models are obtained. The self model is predominantly completed by the intelligent system designer and can be regarded as an abstract description of the underlying system; it only needs to represent those features that are relevant to the ARCHON layer. The models of acquaintances, on the other hand, need not be so detailed as they are used for co-operative purposes rather than for detailed control.

The planning and co-ordination module (PCM) represents the main reflective part of the ARCHON layer. If an exception occurs, it is the task of the PCM to reason about it and find a way out. Its influence on the monitor is mainly through the interface to the behaviours that are currently being executed.

One means of verifying the general architecture is to ensure that it fulfils the requirements that emerged from the operator analogy. As Fig. 3 shows, this exercise ensures that the ARCHON agent architecture provides all the desired functionality to support both local control and co-operative interactions.

4 Implementing the ARCHON layer modules

Having identified the basic functional architecture, a number of important choices need to be made about the realisation of the individual modules. Two crucial, but

opposing, design issues are needed to make the implementation *general* enough to be usable in a number of different application domains, yet *powerful* enough to meet the real-time performance characteristics demanded by the underlying industrial applications. There are no universally appropriate techniques that provide both of these characteristics; those which lead to powerful systems are often hand-crafted for a given problem, whereas those which are appropriate to a range of problems are often too slow to meet the real-time performance requirements of the application.

To circumvent this problem, it was decided to adopt a hybrid design; one with facilities to meet real-time needs and facilities which enable the framework to be used in a broad class of applications. In devising this hybrid, it was first necessary to identify those techniques which provide an application with power (adapted) [19]:

- specialisation into modalities; by identifying functions which need to be performed and developing specialised subsystems to implement them, the overall system performance can be enhanced and a clear modular architecture produced.
- compilation of knowledge and behaviour; by compiling knowledge and reasoning mechanisms, processing power can be greatly improved. Rather than having to reason about which actions should be taken, an agent merely has to recognise the situation and carry out the associated action sequence.

Those techniques which enhance generality are

□ explicit representations; explicit representations of the world and an agent's actions allow high-level descriptions of agent behaviour to be formulated. Such mechanisms enable it to operate in a wide variety of situations as it can embody general principles, rather than be tightly coupled to the specific situation (cf. precompiled mechanisms).

□ generic structures; by using structures which are meaningful in many different situations to represent domain-specific information, the associated reasoning mechanisms can obtain a degree of generality; base the behaviour on the structure imposed by the generic representation mechanism, rather than the idiosyncrasies of the problem being tackled.

These principles were used to guide ARCHON's implementation architecture. First, the modular functional architecture was used as a direct guide for the implementation architecture, and so each module is responsible for a particular modality; the PCM is responsible for assessing the global situation and for dealing with issues involving other agents, the monitor for dealing with local control activities, the AIM is responsible for management of exchanged information, the HLCM with communication etc.

Secondly, the agent models and the information management facilities were devised to be reusable in a number of different applications. These structures obviously need to be instantiated for a given application (e.g. an agent's specific skills need to be described, but the notion that agents have skills is generic and can be reasoned about in a domain-independent manner).

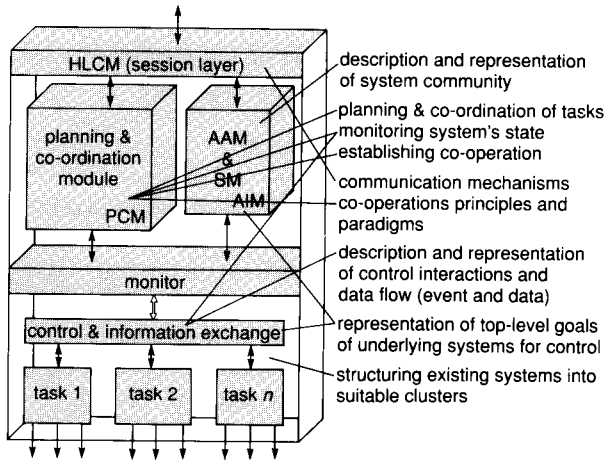

Fig. 3 Requirements mapped into the architecture

The need for both compiled behaviour and explicit representations was another key factor when making decisions about the implementation philosophy of the main architectural components. Compiled behaviour means agents can react to changing situations very quickly as they do not reason about the world. Using explicit representation of the world and encoding knowledge in a declarative manner, on the other hand, typically leads to the opposite properties; the implementation can be more decoupled from the environment, but it often lacks the speed to handle complex reasoning tasks in unconstrained domains. Matching these properties against ARCHON's requirements leads to the following implementation decisions.

- Planning and co-ordination (PCM); the number of co-operative requests occurring will be relatively few in number because agents will spend most of their time engaged in problem-solving, rather than in communicating information, and are typically capable of solving a substantial proportion of their problems themselves. It is also fairly easy to identify the types of co-operation request which may occur and the reasons behind them. For example, agents may request pieces of information or ask for tasks to be performed because they cannot be realised locally or because it is deemed better to ask for somebody else to carry them out. As such activities can be described at a relatively high (general) level and because they are relatively few in number, such knowledge is well suited to a declarative reasoning mechanism with explicit representation of the domain of co-operation.
- Local control (monitor); requires a fast response to numerous events and situations arising in the domain system. Such control varies from application to application and is difficult to generalise. Therefore, it is well suited to the compiled action approach.
- Situation assessment (PCM and Monitor together); assessment activities are intermediate in number and in terms of their generality. Some high-level assessment functionality can be considered general, whereas other components need to be specialised for a particular application. Therefore, in terms of knowledge representation, this functionality has been implemented as a mixture of declarative rules and precompiled mecha-

nisms.

We provide below the details of the implementation of the HLCM, AIM, PCM and the monitor.

4.1 High-level communication

The functionality of the HLCM allows agents to establish meaningful dialogues necessary for decentralised problem-solving and co-ordination. The HLCM provides both generic communication functionality (e.g. 'send this message to all agents which are interested in it') and physical communication functionality (e.g. classical send and receive to a known agent). The three key services provided by the HLCM are intelligent addressing, filtering and message scheduling.

□ Intelligent addressing allows agents to send messages to 'relevant' acquaintances. The relevance of a message for an acquaintance is determined using parameters provided by the PCM and information stored in the acquaintance models.
□ Filtering allows agents to receive only relevant messages. For example an agent can use filtering facilities to receive messages only from certain acquaintances or about certain objects.
□ Message scheduling allows agents to influence the order in which the messages are processed. This ordering mechanism is based on priorities. The scheduling mechanism also supports a time-out facility, e.g. messages will become obsolete after a certain period of time.

With respect to the ISO/OSI standard, the functionality of the session layer has been integrated into the HLCM. This session layer functionality provides a standardised set of communication services based on the broadly used communication protocol TCP/IP. To this end, it creates various session entities based on TCP/IP which employ protocols in order to implement the services provided to session users. The session layer functionality is continuously checking the communication links and provides automatic recovery of connection breaks if possible. Furthermore, it provides statistical information about the message traffic handled.

4.2 AIM

The complexity of the information that needs to be exchanged in industrial applications is substantially greater than in more traditional DAI applications. For example, in electricity management we can easily find data sets of several Mbytes in size (e.g. the updated topology of a section of the network containing all elements with their current status). A number of agents may be interested in certain aspects of such data, but rarely in all of it. Therefore, either the agent where this information was generated sends the complete set to all interested agents (so that they can select whatever they want and discard the rest), or the agent could select all subsets that may be required by some other agents. Both approaches are unsatisfactory; the first because it would overload the communication channels with irrelevant information, and the latter because it would consume considerable computation resources for results that may not be needed in the end. Thus, the solution developed for ARCHON consists of a distributed (federated) database architecture embedded in the general communication facilities of ARCHON (within the AIM module)†. The main idea in this approach is that any generated information is stored and kept at the source, i.e. in the AIM module of the generating agent.

The architecture of the AIM module is based on three main components:

- a common object-oriented database model, the 3DIS.
- a common database language, the 3DIS/ISL.
- a co-operation architecture, called the distributed AIM.

In the AIM module of each ARCHON agent, a 'local schema', one or more 'export schemas', a number of 'import schemas' and an 'integrated schema' have to be developed. Each of these schemas is represented in the 3DIS information modelling formalism.

Each schema represents the classification and organisation defined on the data, is described by a directed acylic graph, and contains the types, subtype/supertype relationships among types, and the mappings and behaviours (operations) defined on each type. The local schema represents the structure of the information that is stored locally in the information system. Export schemas represent the structure of the information an agent wishes to share with other agents. Each import schema represents the structure of the information that this agent wishes to access and another agent is willing to share. Therefore, one agent's import schema is exactly the same as another agent's export schema. The integrated schema represents a coherent view of all the information that an agent can access, including both the local information and a part of the remote information which is modelled in its import schema and the agent wishes to access.

Within an agent, the specification of the relationships between the various schemas is done with the use of (type and map) derivation operators, i.e. export schemas are derived from the local schema. The integrated schema of an agent is also derived from the local schema and the import schemas of the agent. Queries that originate within an agent (local queries) can be evaluated against any of the schemas in an agent, but the default is the integrated schema. Queries that come from another agent (remote queries) are always evaluated against the appropriate export schema. On the basis of the derivation specification, the query processor of an agent decomposes queries on the integrated schema in terms of queries on the local schema and queries on relevant import schemas of another agent. The query processor at another agent, in turn, processes remote queries in terms of its appropriate export schemas and transforms them into queries on its local schema.

† A centralised control and a common structure (e.g. a global shared schema) describing the information shared among different application activities is too restrictive, unnecessarily constrains the freedom of each agent, and certainly in large organisations, leads to cumbersome and bureaucratic maintenance procedures. Additionally, a centralised approach cannot be accomplished efficiently if there is a large number of agents.

5 Example: co-operative diagnosis in electrical networks

Alarms analysis in electrical networks for many years has been one of the major problem domains for the application of artificial intelligence in the energy management industries. The example that has been selected to show the instantiation process of the multi-agent system involves several fault diagnosis systems and is available in the dispatching control room of IBERDROLA in Bilbao, Spain.

The increasing automation and complexity of the automatic controllers have brought the electric utility to the point where human intervention is scarcely needed, but whenever it does occur, the responsibility of the operator's decisions is greater than ever before. This increase in automation has also produced an increase in the amount, reliability and complexity of the information received. In order to help the operator during the monitoring of the network, the necessity of providing an abstract view of the situation has arisen. This, in turn, has led to the development of several expert systems in a multi-agent community to help during this process (Fig. 5):

- alarms analysis agent (AAA); the objective of this agent is to analyse the non-chronological alarm messages in order to identify the element at fault. The time tag attached to these messages refers to the arrival time but not to the time of the fault. Thus it has to be treated with caution.
- breakers and relays supervisor (BRS): the objective of this agent is similar to that of the AAA, but the analysis is based on chronological alarm messages, whose time tag is stamped locally at the substation, allowing a correct sequencing of events. However, these data are transmitted with a delay due to a lower priority in relation to other data.
- black-out area identifier (BAI): the objective of this system is to identify the section of the network that has initially been isolated at the time of the occurrence of the fault, and before the automatic reclosing trial mechanisms started to restore the network. The element at fault has to be within that area and, if all the automatic mechanisms work correctly, they isolate the faulty element. This agent works with non-chronological alarm messages and snapshots.
- control system interface (CSI); this agent acts as a front end between the control computer and the other three agents that are in charge of the diagnosis process.

An important example of co-operation in this system involves the information interchange between the AAA, BRS and BAI agents. The AAA and the BRS have to produce the same result from different sources of information, and the BAI applies different knowledge to this information producing a result that should be coherent with the AAA and the BRS results.

Let us assume that a block of non-chronological alarm messages has been provided by the SCADA (system control acquisition data), and these alarm messages have been identified as related to a disturbance by the task Get_Alarm_Messages of the CSI. This information is received at the CSI monitor level through the corresponding MUs. The monitor sends this information to the PCM as intermediate results. When the PCM receives

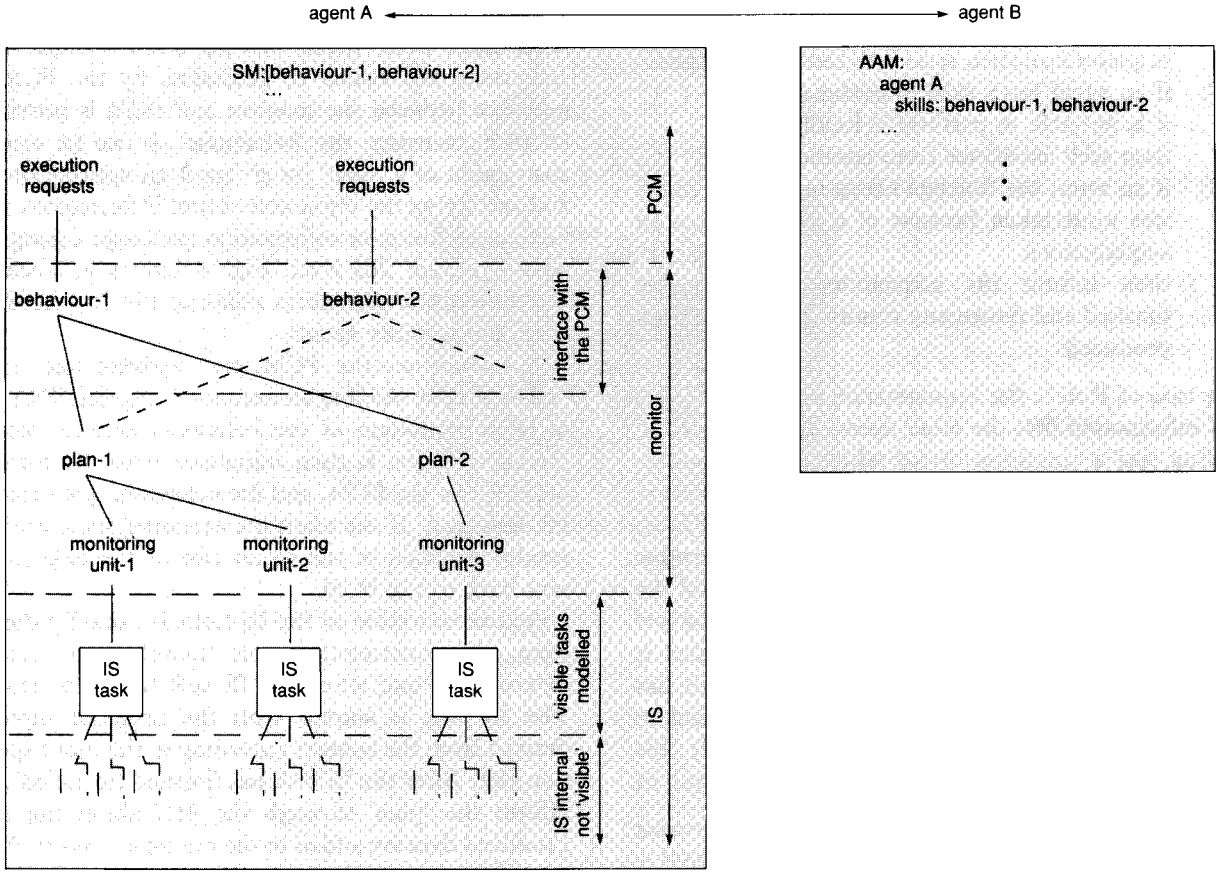

Fig. 4 Modelling control in the monitor

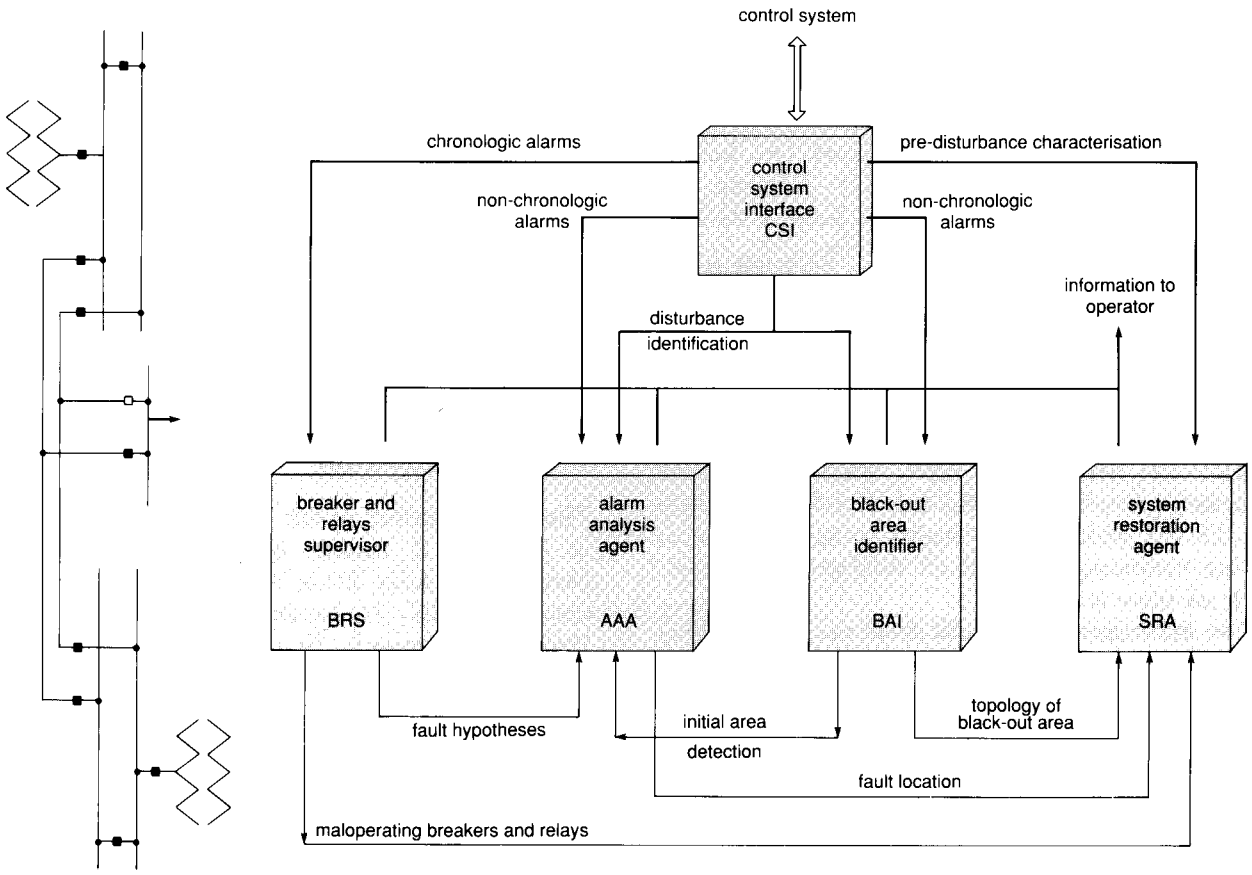

Fig. 5 Agents in the IBERDROLA application

this information, it is passed to those agents with a declared *interest* in it. In our example, this information is sent to the AAA and BAI agents:

ALARM_MESSAGES,
DISTURBANCE_IDENTIFIER--->AAA,BAI

Some time later, the same process is repeated when the corresponding chronological alarms are provided by the SCADA system. In this case, the BRS has declared an *interest* in these data:

ALARM_MESSAGES--->BRS

At this point of the execution, the three agents (AAA, BAI and BRS) are working in parallel.

When the AAA receives the alarm messages and the corresponding disturbance identifier has identified the alarm messages as a consequence of a fault, the 'behaviour' New_Alarms is triggered and its associated plan executed. The plan execution activates the MU Set_Alarm_Messages, which provides the IS with the alarm messages. Once this MU finishes, this plan has been completed, the 'behaviour' New_Fault is triggered, and the execution of its associated plan generates a set of hypotheses.

At the same time, the BAI has received the same information (alarm messages and disturbance identifier), which triggers the 'behaviour' Initial_Black_Out_Area. The execution of the associated plan activates sequentially several MUs, e.g. the Initial_Area_Out_Of_Service MU, which provides the Initial_Area_Out_Of_Service data. When the plan is completed, this information is sent to the PCM as a final result of the behaviour. The BAI then checks whether there is any agent that is interested in this information and finds out that the AAA is interested in it. Consequently, the Initial_Black_Out_Area is sent to the AAA:

INITIAL BLACK OUT AREA--->AAA

Simultaneously, but with a certain delay, the BRS agent has started working on the analysis of the chronological alarm messages, because as soon as its PCM has received the chronological alarm messages, the New_Block behaviour is triggered. Its associated plan is executed and the Read_Chronological_Alarm_Messages MU is triggered, which means that the alarm messages are fed into the IS and that the block of chronological alarm messages is divided into smaller subsets of 1 s. However, for simplicity, we assume that there is just one subset. Once this MU finishes, the plan is also completed and the behaviour Subset_Analysis is executed. The Subset_Alarm_Messages MU identifies a faulty pattern in the alarms subset received, and consequently the Trips_Generation and Hypotheses_Generation MUs are executed. As results of the execution of the last MU, the monitor receives the Generated_Hypotheses, which are sent to the PCM as an intermediate result. The BRS then checks whether there is any agent interested in this information and finds out that the AAA is interested in it. Consequently, the Generated_Hypotheses data is sent to the AAA:

GENERATED HYPOTHESES->AAA

The AAA, after the New_Fault behaviour execution has

finished, continues with the alarm messages analysis. In the meantime, the other two agents proceed with their analysis. The following situations may then occur.

□ The Initial–Black–Out–Area is available to the AAA. This triggers the Refinement_Based_On_Initial_Black_Out_Area behaviour, possibly reducing the number of hypotheses to be validated because the BAI has given a focused view of the situation.
□ The Generated_Hypotheses provided by the BRS are available and sent to the AAA, which can now trigger the Refinement_Based_On_Generated_Hypotheses behaviour, obtaining a better reordering of the hypotheses to be validated and a benefit in finding the element at fault.
□ The Validated Hypotheses provided by the BRS are available and sent to the AAA, which triggers the Refinement_Based_On_Validated_Hypotheses behaviour; this has the same functionality as the previous one, but the reordering is based on validated hypotheses, which are more accurate.
□ If no information is available from the BAI or BRS, the AAA proceeds with the validation of the hypotheses as a stand-alone agent. Therefore, if the other agents are down or they are too slow to provide the information, the AAA continues and finds a faulty element, although its reliability is less and the speed in finding the solution is reduced.

6 Conclusions and future work

The view presented here is only partial, and interested readers should refer elsewhere [20] for further details. We have identified heterogeneity and the ability of an operator to actively participate in problem-solving as critical design forces in the construction of co-operation frameworks for industrial applications. The type of problems that ARCHON is addressing have been detailed, and the associated requirements have been mapped into the functional architecture.

Finally, ARCHON's hybrid approach and the use of generic structures have been identified as key techniques for confronting the power generality problem.

The Project has applied these concepts to applications in order to achieve its final goal; the ARCHON framework as an industrial product for a wide range of supervision and control applications. This involves a two-step approach; first, to restructure the control level of already existing applications and, at the same time, develop the 'principle ARCHON layer'. This refers mainly to designing and implementing the monitor, which finally deals with the intelligent systems of the application and, by its very nature, is highly domain-dependent.

As mentioned in Section 1, the ARCHON architecture as it now stands only concentrates on loose coupling of semi-autonomous agents. Clearly, if an organisation has a collection of pre-existing computational systems, each dealing with a separate aspect of the same underlying domain, then the architecture presents an opportunity for bringing these together into a useful co-operative framework. Pilot studies carried out within the Project itself provide a convincing demonstration of the use of ARCHON where there are pre-existing computational systems. However, ARCHON is not designed only for pre-existing systems, but also providing co-operation between any set of semi-autonomous systems. Clearly, future work needs to be carried out to examine the use of ARCHON concepts in a more general case. Before this can be done, an important matter needs to be considered; ARCHON, on its own, would restrict an integration approach to formulating a solution only in terms of loosely coupled semi-autonomous agents. ARCHON concepts may well have their best use as enhancements to more conventional (e.g. client-server or any other distributed computing platform) integration architectures.

7 Acknowledgments

The work described in this Project has been carried out in ESPRIT II Project ARCHON (P-2256), whose partners are Atlas Elektronik, Germany; Amber SA, Greece; CERN, Switzerland; EA Technology, UK; Framentec-Cognitech, France; Iberdrola, Spain; JRC Ispra, Italy; Labein, Spain; Queen Mary and Westfield College, UK; University of Amsterdam – FWI, Netherlands; Technical University Athens – CNRG, Greece; University Brussels – IRIDIA, Belgium; University of Porto – Robotics Lab, Portugal; Volmac, Netherlands.

In writing this paper, we are acting on behalf of the whole consortium by disseminating the project's results; we are not claiming to have conceived or implemented all of the concepts which are described herein. Although these concepts originated from interactions between all of the consortium's members it is nevertheless possible to identify those individuals who have made significant contributions to certain aspects of the project: architecture design (Thies Wittig, Abe Mamdani, Erick Gaussens), the monitor (Erick Gaussens, Daniel Gureghian, Jean-Marc Loingtier, Bernard Burg), PCM (Nick Jennings, Jeff Pople, Jochen Ehlers), AIM (Frank Tuijnman, Hamideh Afsarmanesh, Giel Wiedijk), HLCM (Claudia Roda, Jutta Müller), the agent models (Nick Jennings), the C++ implementation (Rob Aarnts) and the electricity management application (Jose Corera and Juan Perez).

The authors would also like to thank Brice Lepape and Patrick Corsi from the CEC. Without their engagement in the definition phase of the project and their close supervision and guidance during the life-time of ARCHON, we would not have achieved these results.

8 References

[1] ARLABOSSE, F., BIERMANN, J., GAUSSENS, E., and WITTIG, T.: 'Industrial control: a challenge for the applications of AI'. Proc. ESPRIT Conf. 1987, Brussels (North-Holland)

[2] WITTIG, T.: 'Power systems fall under KRITIC's eye' *In*: 'Modern power systems' (United Trade Press, London, 1987) Vol. 7(1)

[3] OLIVEIRA, E., CAMACHO, R., and RAMOS, C.: 'A multi-agent environment in robotics', *Robotica, Int. J. Inform. Educ. Res. Robot. Artif. Intell.*, 1991, **9**, (4), pp. 431–440

precalciner flash furnace'. Proc. IEEE Conf. on Application of Adaptive & Multivariable Systems, Hull, UK, 1982, pp. 56–59
[5] KING, R. E., and KARONIS, F. C.: 'Multi-level expert control of a large scale industrial process' GUPTA, M. M., and YAMAKAWA, T. (Eds.) *in*: 'Fuzzy computing' (North Holland Co., 1988)
[6] MALANDAIN, E., and SKAREK, P.: 'An expert system for accelerator fault diagnosis'. Proc. IEEE Conf. on Particle Accelerators, Washington DC, 1987, Vol. I, p. 559
[7] BOND, A. H., and GASSER, L.: 'Readings in distributed artificial intelligence' (Morgan Kaufmann, 1988)
[8] GASSER, L., and HUHNS, M. N.: 'Distributed artificial intelligence Vol. II' (Pitman, 1990)
[9] HUHNS, M. N.: 'Distributed artificial intelligence' (Pitman, 1989)
[10] JENNINGS, N. R.: 'Cooperation in industrial systems'. Proc. ESPRIT Conf. Brussels, 1991
[11] AFSARMANESH, H., and TUIJNMAN, F.: 'ISA: an architecture to support sharing and exchange of information among ARCHON agents'. Technical Report CS-91-07, University of Amsterdam, 1991
[12] RODA, C., JENNINGS, N. R., and MAMDANI, E. H.: 'The impact of heterogeneity on cooperating agents'. Proc. AAAI Workshop on Cooperation among Heterogeneous Intelligent Systems, Anaheim, Los Angeles, 1991
[13] GAUSSENS, E. J.: 'Needs and opportunities for expert systems in process control'. Vacation School for Process Control, University of Strathclyde, UK, 1990
[14] LEVESON, N. G.: 'The challenge of building process control software', *IEEE Softw.*, 1990, pp. 55–62
[15] HALL, L. E., AVOURIS, N. M., and CROSS, A. D.: 'Interface design issues for cooperating expert systems'. Proc. 10th Int. Conf. in Expert Systems, Avignon, 1990, pp. 455–469
[16] STEINER, D. D., MAHLING, D. E., and HAUGENEDER, H.: 'Human computer cooperative work'. Proc. 10th Workshop on Distributed Artificial Intelligence, Texas, 1990
[17] RODA, C., JENNINGS, N. R., and MAMDANI, E. H.: 'ARCHON: a cooperation framework for industrial process control', in DEEN, S. M. (Ed.) 'Cooperating knowledge based systems', (Springer-Verlag, 1990), pp. 95–112
[18] AFSARMANESH, H., TUIJNMAN, F., WIEDIJK, M., and HERTZBERGER, L. O.: 'Distributed schema management in a cooperation network of autonomous agents'. Proc. 4th IEEE Int. Conf. on Database and Expert Systems Applications, 1993, Springer Verlag, *Lect. Notes Comput. Sci.*, **720**
[19] KISS, G.: 'Variable coupling of agents to their environment: combining situated and symbolic automata'. Proc. MAAMAW, Kaiserslautern, Germany, 1991
[20] WITTIG, T. (Ed.): 'ARCHON: an architecture for multi-agent systems' (Ellis Horwood, Chichester, 1992)

The paper was first received 6 June and in revised form 13 July 1994.

T. Wittig is with Atlas Elektronik GmbH, TEF3, Sebaldsbrücker Heerstrasse 235, 28305 Bremen, Germany; N. R. Jennings and E. H. Mamdani are with the Department of Electronic Engineering, Queen Mary and Westfield College, University of London, Mile End Road, London E1 4NS, UK.