THE ARCHON™ SYSTEM AND ITS APPLICATIONS

N. R. Jennings

Department of Electronic Engineering,
Queen Mary and Westfield College,
University of London,
Mile End Road,
London E1 4NS, UK.
email: N.R.Jennings@qmw.ac.uk

ABSTRACT

ARCHON™ (ARchitecture for Cooperative Heterogeneous ON-line systems) was Europe's largest project in the area of Distributed Artificial Intelligence (DAI). It devised a general-purpose architecture, software framework, and methodology which has been used to support the development of DAI systems in a number of real world industrial domains. Some examples of the applications to which it has been successfully applied include: electricity distribution and supply, electricity transmission and distribution, control of a cement kiln complex, control of a particle accelerator, and control of a robotics application. The type of cooperating community that it supports has a decentralised control regime and individual problem solving agents which are large grain, loosely coupled, and semi-autonomous.

This paper will tackle a broad range of issues related to the application of ARCHON technology to industrial applications. Firstly, it gives the rationale for a DAI approach to industrial applications and highlights the characteristics which typify this important domain. Secondly, the ARCHON framework is detailed - with a special emphasis being placed upon the implementation architecture. Thirdly, a brief resumee and status report of the main applications is presented. Finally, the lessons learned and the future plans are presented.

1. INTRODUCTION

In many industrial applications a substantial amount of time, effort and finance has been devoted to developing complex and sophisticated software systems. These systems are often viewed in a piecemeal manner as isolated islands of automation, when, in reality, they should be seen as components of a much larger overall activity (Jennings, 1994). The benefit of taking a holistic perspective is that the partial subsystems can be integrated into a coherent community in which they work together to better meet the needs of the entire application. By the very fact that they are integrated, the finite budgets available for information technology development can be made to go further - agents can share a consistent and up-to-date version of the data, basic functionalities need only be implemented in one place, problem solving can make use of timely information which might not otherwise be available, and so on.

Two components are required to devise a well-structured integrated community: a framework which provides assistance for interaction between the constituent subcomponents and a methodology which provides a means of structuring interactions. ARCHON addresses both of these facets: providing a decentralised software framework for creating Distributed AI (DAI) systems for industrial applications and devising a methodology which offers guidance on how to decompose an application to best fit with the ARCHON approach (Wittig, 1992). The former is concerned with providing the necessary control and level of integration to help the subcomponents to work together; the latter is concerned with decomposing the overall application goal(s) and with distributing the constituent tasks throughout the community.

ARCHON's individual problem solving entities are called agents; these agents have the ability to control their own problem solving and to interact with other community members. The interactions typically involve agents cooperating and communicating with one another in order to enhance their individual problem solving and to better solve the overall application problem. Each agent consists of an *ARCHON Layer* (AL) and an application program (known as an *Intelligent System* (IS)). Clearly distinguishing between an agent's social know-how and its domain-level problem solving means that the ARCHON approach is both flexible and open - imposing relatively few constraints on the application designer and yet providing many useful facilities. Purpose-built ISs can make use of the ARCHON functionality to enhance their problem solving and to improve their robustness. However pre-existing ISs can also be incorporated, with a little adaptation, and can experience similar benefits - see Jennings (1994) for a detailed account of how this process was carried out for the particle accelerator control application. This latter point is important because in many cases developing the entire application afresh would be considered too expensive or too large a change away from proven technology (Jennings and Wittig, 1992).

To successfully incorporate both purpose-built and pre-existing systems, community design must be carried out from two different perspectives simultaneously. A top down approach is needed to look at the overall needs of the application and a bottom up approach is needed to look at the capabilities of the existing systems. Once the gap between what is required and what is available has been identified the system designer can choose to provide the additional functionality through new systems, through additions to the existing systems, or through the ARCHON software itself. This methodology shapes the design process by providing guidelines for problem decomposition and distribution which reduce inefficiencies and is described more thoroughly in Varga *et al*. (1994) and Cockburn and Jennings (1995).

This paper is organised along the following lines: section two provides a detailed view of ARCHON's software framework - covering both inter-agent interactions and interactions between the AL and its underlying IS. Section three provides an overview of ARCHON's applications, concentrating mainly on the electricity distribution and the electricity transmission systems. Finally, section four presents the conclusions of this work and outlines some future plans.

2. THE ARCHON ARCHITECTURE

The ARCHON software has been used to integrate a wide variety of application program types under the general assumption that the ensuing agents will be loosely coupled and semi-autonomous. The agents are loosely coupled since the number of interdependencies between their respective ISs are kept to a minimum; the agents are semi-autonomous since their control regime is decentralised (meaning each individual decides which tasks to execute in which order). The ISs themselves can be heterogeneous - in terms of their programming language, their algorithm, their problem solving paradigm, and their hardware platform (Roda *et al.*, 1991) - as their differences are masked by a standard AL-IS interface. An AL views its IS in a purely functional manner - it expects to invoke functions (*tasks*) which return results - and there is a fixed language for controlling this interaction.

In an ARCHON community there is no centrally located global authority, each agent controls its own IS and mediates its own interactions with other agents. The system's overall objective is expressed in the separate local goals of each community member. Because the agents' goals are usually interrelated, social interactions are required to meet global constraints and provide the necessary services and information. Such interactions are controlled by the agent's AL; relevant examples include: asking for information from acquaintances, requesting processing services from acquaintances, and spontaneously volunteering information which is believed to be relevant to others.

In more detail, an agent's AL needs to: control tasks within its local IS, decide when to interact with other agents (for which it needs to model the capabilities of its own IS and the ISs of the other agents), and communicate with the others. These basic requirements are embodied in ARCHON's modular and layered implementation architecture (fig 1). This architecture contains four modules - Monitor (section 2.1), Planning and Coordination Module (PCM) (section 2.2), Agent Information Management (AIM) module (section 2.3), and High Level Communication Module (section 2.4)

In terms of its hardware and software requirements, the ARCHON layer was originally implemented using LISP; later it was ported to C++ and now runs under UNIX on Sun-4 architecture workstations and under Linux on i386 machines.

2.1 Monitor

The Monitor is responsible for controlling the local IS. Each IS task is represented in the Monitor by a **monitoring unit** (MU). MUs present a standard interface to the Monitor whatever the host programming language and hardware platform of the underlying IS. Figure 2 shows a graphical representation of an MU called `TransformTelemetry` which takes `TELEMETRY` as an input and produces `TRANSFORMED-TELEMETRY` as an output. The IS task associated with this MU is called `TRANSFORM-TELEMETRY` in the ARCHON layer.

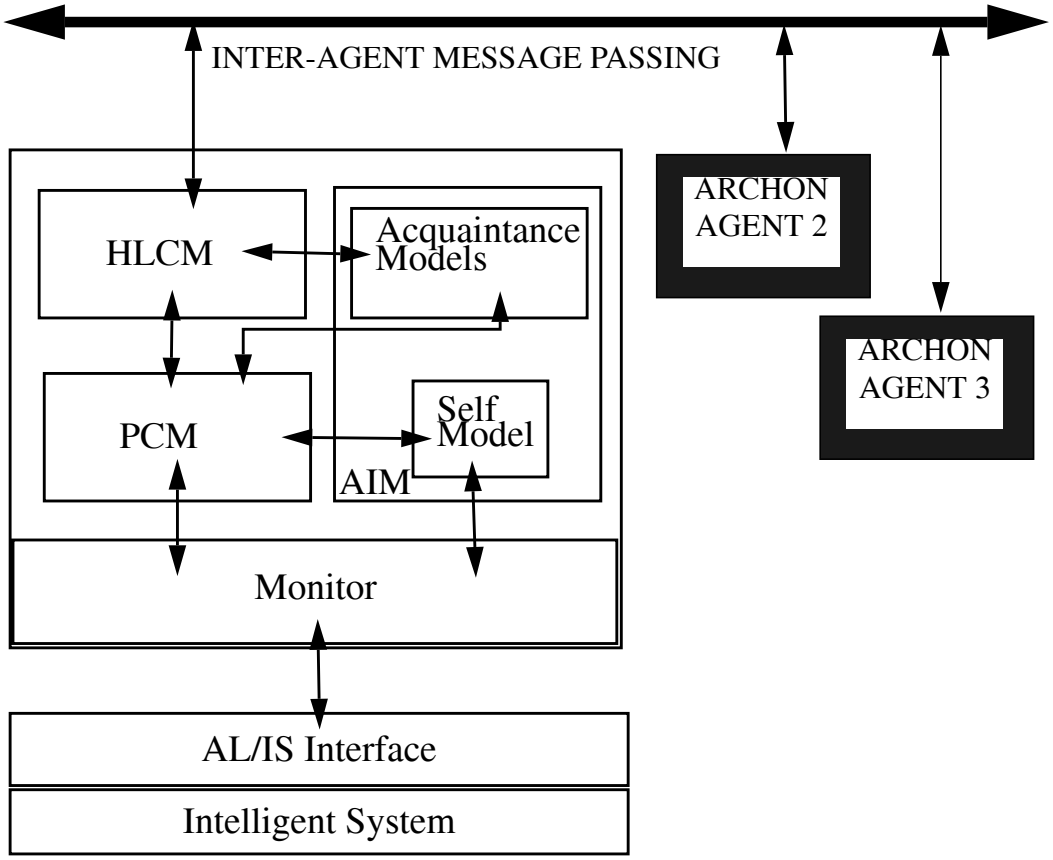

Figure 1: Structure of an ARCHON Agent and an ARCHON Community

MUs can send and receive messages (control, confirmations and requests) to/from the IS. All messages have to pass through the AL-IS interface which performs the translation and interpretation required for the IS to understand the AL directives and for the AL to understand the IS messages. For instance, in the above example, the interface functionality involves invoking the IS function `transform_telemetry`, passing the arguments in the form in which they are expected by the IS's host language, and returning the transformed telemetry in the expected format.

For the IS to be able to react to an AL directive the interface must translate the command into the corresponding local control action(s). However this interpretation is subject to the capabilities of the IS - for example, `KILL` in a C program may just mean that - kill a process; whereas in a rule-based Prolog system it may mean clear all the facts asserted by the current task and then stop. This means that the interface has to be specialised to the IS's programming language - although the 'C' language version can be used as foreign code in other programming languages. Other interface functionalities include specifying how many invocations of a particular task can run in parallel and how many can be queued should that limit be reached. For example, it may be possible to run only one invocation at a time (e.g. updating a global data store), or a task can have multiple instantiations running in parallel but it may be desirable to have a limit and to queue other requests for that task until one has completed.

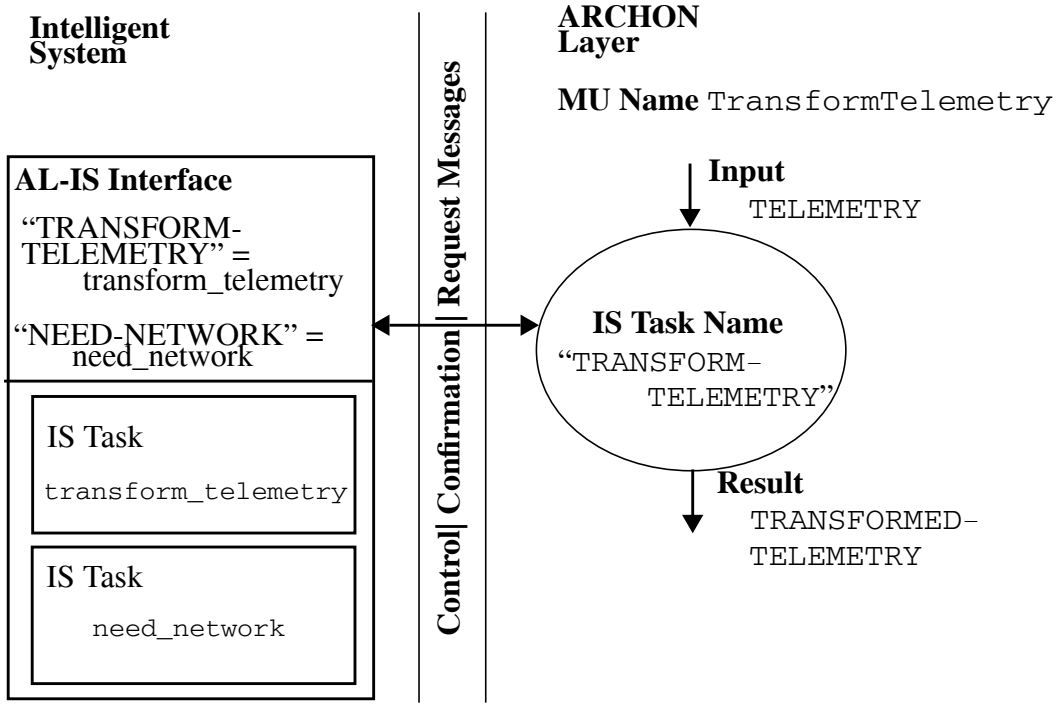

Figure 2: TransformTelemetry Monitoring Unit

MUs represent the finest level of control in the ARCHON layer, at the next level of granularity there are **plans**. Plans are acyclic OR-graphs in which the nodes are MUs and the arcs are conditions. These conditions can: be dependant on data already available from previously executed MUs in the plan; be dependant on data input to the plan when it started; make use of the locking mechanism for part of the plan; and be used to return intermediate results before a plan has completed.

A sample plan which ensures that the electricity network is loaded, before fault diagnosis can commence, is shown in figure 3. Firstly, the `TELEMETRY` is transformed into a suitable format by the `TransformTelemetry` MU. Secondly, the MU `Need-Network` checks whether that portion of the network from which the telemetry originated is available locally. Next there is a decision point: if a `network_descriptor` was returned by the previous MU then the relevant portion of the network needs to be obtained, if not then the network is already available locally and the plan ends. Assuming that the network needs to be loaded, an appropriate request will be sent to the PCM and eventually the network description should arrive. On return of the network the `CreateNetworkModel` MU is executed and the plan terminates.

The highest level at which the IS's activities are represented is the behaviour level. **Behaviours** contain a plan, a trigger condition for activating the behaviour, descriptions of the inputs needed by the activity and the results which will be produced, and any children of the behaviour. There are two types of behaviour: those that are visible

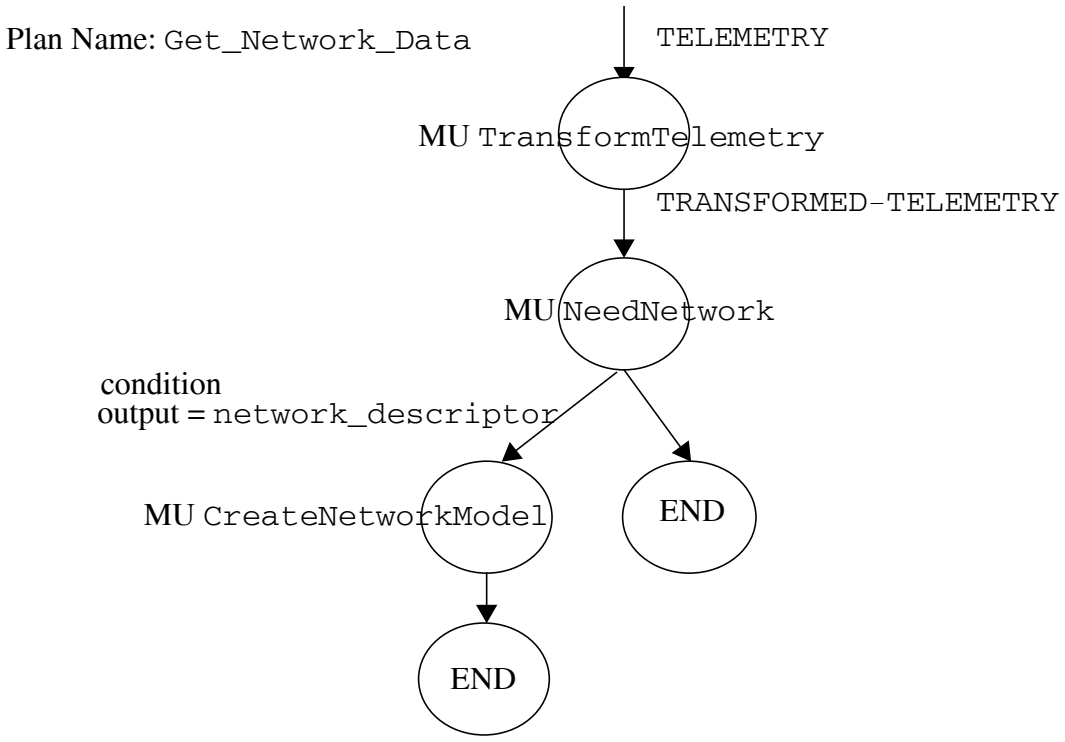

Figure 3: Get_Network_Data Plan

to the PCM (and the other AL components) and those that are purely internal to the Monitor (e.g. `DiagnoseFault` in figure 4). The former type are called **skills** (e.g. `DealWithTelemetry` in figure 4) and they may be triggered by data arriving from other agents or by direct requests from other agents.

Once the PCM activates a skill the Monitor is responsible for its execution. This involves testing any plan conditions, activating MUs as and when appropriate, and dealing with messages from the IS. In order to activate an MU the Monitor ensures that the necessary inputs are present - they could be available because they were an input to the behaviour in the first place or because they were generated by the execution of an earlier MU in the same plan. However if a piece of information is not available in the local context then a request is forwarded to the PCM (e.g. in the `Get_Network_Data` plan a request for the network associated with the generated descriptor may not be available in the local context). Finally, the Monitor passes the skill's results back to the PCM and then activates any child behaviours.

2.2 Planning and Coordination Module

The PCM is the reflective part of the ARCHON Layer, reasoning about the agent's role in terms of the wider cooperating community (Jennings and Pople, 1993). This module has to assess the agent's current status and decide which actions should be taken in order to exploit interactions with others whilst ensuring that the agent contributes to the community's overall well being. Specific examples of the PCM's functionality in-

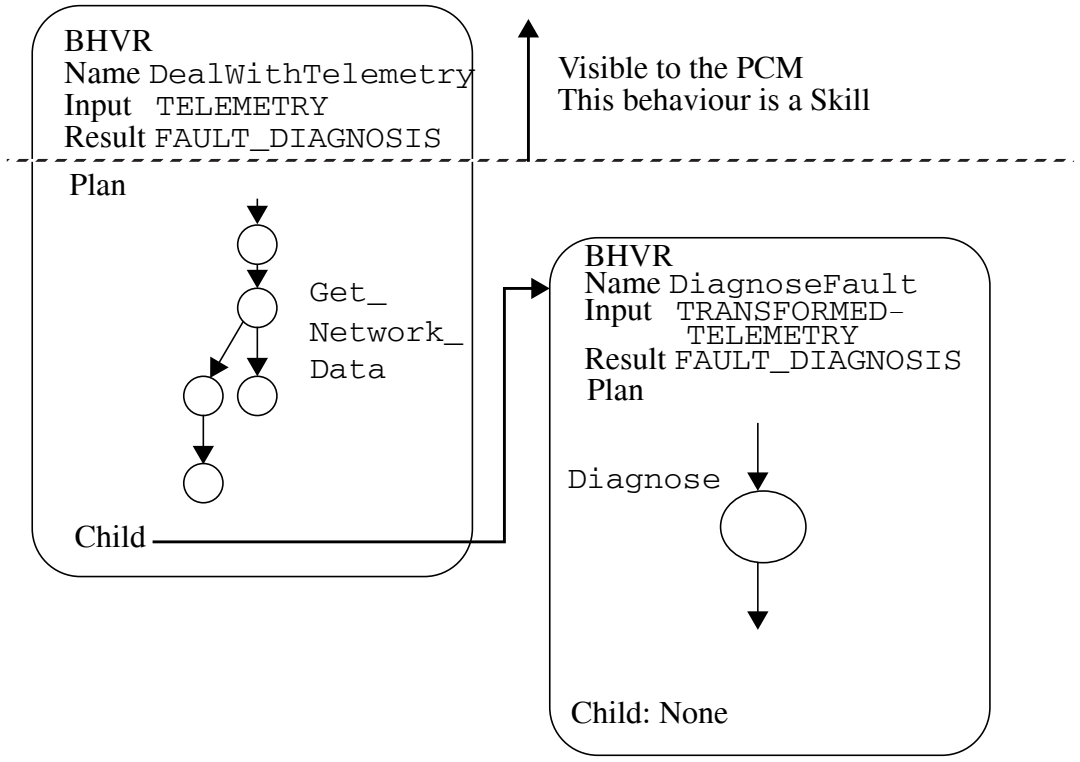

Figure 4: DealWithTelemetry skill with DiagnoseFault child behaviour

clude: deciding which skills should be executed locally and which should be delegated to others, directing requests for cooperation to appropriate agents, determining how to respond to requests from other agents, and identifying when to disseminate timely information to acquaintances who would benefit from receiving it.

The PCM is composed of generic rules about cooperation and situation assessment which are applicable in all industrial applications - all the domain specific information needed to define individual behaviour is stored in the self and acquaintance models. The former contains information about the local IS and the latter contains information about the other agents in the system. The type of information contained in both models is approximately the same, although it varies in the level of detail, and includes the agent's skills, interests, current status, workload and so on. For example, in order to determine how to process the request to provide the relevant portion of the network which arises from the `DealWithTelemetry` skill, the PCM will make reference to its self model to see if the information can be provided locally. If the information cannot be provided locally then the acquaintance models are checked to see if another community member can provide it. A second illustration of the interplay between the agent models and the PCM occurs when the Monitor provides the results of a recently completed skill: firstly, the PCM checks the self model to see if the data can be used locally and then it examines its acquaintance models to see if any other agents are believed to be interested in receiving the data. The final major role of the PCM is to deal

with requests arriving from other agents. By reference to its self model, it will decide whether to honour the request and will then activate the necessary skill to provide the requested data; when the information is available it will ensure that a reply is directed to the source of the request.

2.3 Agent Information Management Module

The AIM module is a distributed object management system which is designed to provide information management services to cooperating agents (Tuijnman and Afsarmanesh, 1993). Within ARCHON, it is used to store the agent models and the domain level data.

The self model contains all the definitions of MUs, plans and behaviours. An agent only models those acquaintances with which it will interact; the models themselves contain the agent's name, the information it is interested in (which the local agent can provide), and the skills it can perform (which the local agent may need). As an illustration of the models, consider an agent which is capable of producing information about TELEMETRY. The interest slots of its acquaintance models would contain those agents who are interested in receiving this information and the conditions under which they are interested. The following portion of the acquaintance model specifies that an agent called AVA is interested in TELEMETRY in all cases, that an agent called LVDA is interested when the operation attribute has value autoreclosure, and that the agents called IA and HVDA are interested only if the operation attribute is equal to open or closed

```
INTEREST-DESCRIPTOR
  INFORMATION-NAME: TELEMETRY
  INFORMATION-CONDITION:
    [("AVA", TRUE);
    ("LVDA", (CONTAIN (TELEMETRY,"OPERATION "Autorec""")));
    ("IA HVDA",
              (OR(CONTAIN(TELEMETRY,"OPERATION "Open""))
                   (CONTAIN(TELEMETRY,"OPERATION "Closed""))));]
```

In many industrial applications the domain level data which the agents need to exchange has a complex internal structure. In ARCHON, this structure is specified and maintained by AIM. For example, TELEMETRY is defined in the following manner:

```
TELEMETRY
    Text          STRINGS      Time         INTEGER
    Text_time     STRINGS      Substation   STRINGS
    Plant         STRINGS      Operation    STRINGS
    Source        STRINGS
```

and a specific instance is as follows: (note this piece of telemetry would be deemed to

be of interest to the AVA and LVDA agents)

```
((Telemetry
(Text 'ALARM 01JAN94 12.00/12.00 SBSTN1 SUBSTATION1
     C1 CIRCUIT BREAKER AUTOREC')
(Time 10874390) (Text_time '01JAN94 12.00/12.00')
(Substation 'SUBSTATION1') (Plant 'SUBSTATION1C1')
(Operation 'AUTOREC') (Source 'ALARM'))
```

Once domain data has been stored in AIM it is possible for the AL's reasoning and control mechanisms to retrieve it. By giving it a definite structure, it is possible to access the named sub-parts (e.g. checking that the attribute operation equals open). If the application designer does not require the AL to access these components then the structure's details need not be given. The scheme is also flexible enough to allow for a halfway house where that part of the data the AL needs to reason about is structured, as above, and that part which is not looked at is left unstructured (e.g. the `text` field above).

2.4 High Level Communication Module

The High Level Communication Module (HLCM) allows agents to communicate with one another using services based on the TCP/IP protocol. The HLCM incorporates the functionality of the ISO/OSI Session Layer which continuously checks communication links and provides automatic recovery of connection breaks when possible. Information can be sent to named agents or to relevant agents (decided by reference to interests registered in the acquaintance models).

3. ARCHON'S APPLICATIONS

The ARCHON software and methodology have been used to develop real-world DAI systems in a number of different industrial domains, these include:

- Electricity distribution and supply (section 3.1)
- Electricity transmission and distribution (section 3.2)
- Control of a cement kiln complex (section 3.3)
- Control of a particle accelerator (section 3.4)
- Control of a flexible robotic cell (section 3.5)

In this paper we briefly describe each of these applications - the systems built vary from large scale project demonstrators (electricity distribution and electricity transmission) which show the scaleability of the adopted approach, to the application studies

(the remainder) which were used to test the functionality and operation of the ARCHON system. In all cases, references to more complete descriptions are provided.

3.1 Electricity Distribution & Supply

(refs: Cockburn, 1993; Cockburn and Jennings, 1995; Varga *et al.*, 1994)

This application (called CIDIM) is being developed as an aid to Control Engineers (CEs) who have the job of ensuring continuity of electricity supply to the customers. The CE's main tasks are to plan and carry out maintenance work safely in coordination with the Field Engineer (FE), to identify faults on the network, and to perform remedial actions to restore supply should this be necessary. The electricity network control system used by the CE allows remote operation of circuit breakers and also reports, via telemetry, automatic switching operations in response to a fault, alarms, and load readings. This system covers the high voltage network and part of the low voltage network; however for the remainder of the low voltage network switching for maintenance purposes is done manually by the FE in contact with the CE and fault diagnosis is based on customer telephone calls reporting a loss of supply. The CE can also make use of information about lightning strikes which may be the cause of a fault and so indicate a good starting point for the FE to look for damaged equipment.

CIDIM will be able to assist the CE by automatically providing services such as fault diagnosis, lightning detection, user driven restoration planning, and automatic rechecking of restoration plans, as well as automatically collating much of the information the CE now does manually by reference to standalone systems. Using ARCHON helps information from conventional knowledge sources (such as databases and a telemetry system) to be shared by a number of agents, thus leading to greater consistency across the application. CIDIM consists of 7 agents: a telemetry agent (TA) which receives telemetry and converts it to a standard format, an information agent (IA) which contains information about the network (both static data - eg connectivity of pieces of plant - and dynamic information - eg state of circuit breakers and switches), a weather watch agent (WWA) which knows when and where lightning strikes occur, a high voltage diagnosis agent which uses telemetry to diagnose the location, time and type of faults on the network, a low voltage diagnosis agent (LVDA) which uses telephone calls and lightning information as well as telemetry to diagnose faults, a switch planning agent which allows the user to plan maintenance work, and an advisor agent (AVA) which acts as the user interface to the entire application. The agents represent a number of different types of IS (conventional programs, databases and expert systems), some of them were pre-existing and some were purpose built within ARCHON.

A sample cooperative scenario for this application occurs when the LVDA diagnoses faults. The LVDA receives telemetry relating to the low voltage portion of the network from the TA and triggers an appropriate skill to deal with it (see figure 4). The first step in this skill is to determine whether the telemetry refers to a known fault or whether it relates to a new fault (see figure 3). Assuming the latter, the LVDA needs to obtain the

status of the relevant portion of the network from the IA and so sends it an explicit request. When the IA returns the pertinent part of the network, the LVDA uses it to hypothesise a number of faults- one of which may be that the problem is caused by a lightning strike. This hypothesis can be confirmed or rejected by sending a request to the WWA. Eventually the LVDA will reach some conclusion and present it to the CE via the AVA. This scenario illustrates how agents can ask one another to perform activities which are either essential (getting the network status) or optional (eg asking about lightning) to their own problem solving.

3.2 Electricity Transmission & Distribution

(refs: Abel *et al.*, 1993; Corera *et al.*, 1993)

This application provides a support system for the operators of an electrical dispatching control room (DCR) in their tasks of analysing the occurrences of disturbances and in planning the service restoration procedures. This system will be installed in the main DCR of IBERDROLA and will be used to control the network for the north and northwest part of Spain. The information acquired from the network updates the status of 20,000 elements - breakers, switches and protective relays - with a polling cycle of 10 seconds, and the value of 2000 analogue measurements - power flows, voltages, frequencies - which are polled every 20 seconds. Management of the electricity network, as performed by the operator in the DCR, consists mainly of topology changes (operations on breakers and switches), generation scheduling and control. However in emergency situations this management becomes very complex due to the large number of constraints which need to be taken into consideration and the insufficient quality of the information arriving at the DCR.

This application is structured as a set of 6 agents which interact with one another in order to diagnose and restore the network after disturbances. The diagnosis subset of this community consists of an alarm analysis agent (AAA) which identifies and classifies faults based on alarms, the breaker and relay supervisor (BRS) which identifies and classifies faults based on chronological information, the black-out area identifier (BAI) which identifies a list of all those elements that are out of service initially, and the control system interface (CSI) which detects the occurrence of a disturbance in order to signal to the others that they should start their activities. The restoration subset of the community consists of two agents: the control system interface for restoration which provides an image of the network's current state and the service restoration agent which prepares the restoration plan.

As an illustration of the cooperation in this application the process of cooperative fault diagnosis will be expanded in more depth. The fact that there is a disturbance in the network is determined by the CSI agent (by analysing the chronological and non-chronological alarm messages of breakers, fault recorders and relays that it receives). If it suspects that there is a fault then it prepares and sends blocks of chronological alarm messages to the BRS and blocks of non-chronological alarm messages to the AAA and the BAI. Upon receipt of these messages, each of the agents start their diag-

nosis activities. The AAA first performs a fast approximation which generates a large number of alternatives and then commences on a detailed and time consuming sequential evaluation of each of them. Meanwhile the BAI generates a list of elements which are in the black out area which it sends to the AAA. The AAA is able to prune its search space for its detailed evaluation by removing any of its initial hypotheses which are not in the black out area as computed by the BAI. There is also scope for interaction between the BRS and the AAA - both are able to generate lists of hypotheses which can focus the other's problem solving and they are also able to check each other's diagnoses as a way of increasing the reliability of the final result. This scenario illustrates how agents working with different data and different problem solving know-how can interact to produce more reliable results more quickly.

3.3 Control of a Cement Kiln Complex

(refs: Lembessis and Antonopoulos, 1993; Stassinopoulos and Lembessis, 1993)

This application study is concerned with a simulation of the high level control of a kiln complex in a cement manufacturing process. The control system has to monitor, and set, a number of highly interdependent parameters (eg material feed rate, temperature, gas flow, etc.) to ensure that the final product (cement) has the desired characteristics. A DAI approach was deemed appropriate because the process itself is highly distributed, because there were a number of identifiable functional units in the existing system which needed to interact in order to optimise the production process, and because the process's complexity meant that the currently existing centralised controller was unable to cope with the application's real time constraints.

The implemented ARCHON community for this application consists of 4 agents which together are responsible for the control of the entire manufacturing process. There is a precalciner agent (PCA) which controls the first phase of the kilning process (in which the majority of the calcium carbonate in the raw material is decomposed into calcium oxide and calcium dioxide); a kilning control agent (KCA) which controls the second phase in which calcification is completed and where the chemical reaction between the calcium oxide and the other oxides form a new compound (clinker) which is the main ingredient of cement; a clinker cooling agent (CCA) which deals with the final phase; and a user interface agent which presents information to the operator and accepts control commands.

As an example of the cooperation involved in this application consider the following scenario which is taken from the precalciner and kiln control phase of the process. The 3 control agents each calculate, and then exchange, the key control parameters for that part of the process which they are monitoring - respectively, precalciner compartment temperature, kilning effectiveness and raw material feed rate for the PCA, KCA and CCA. Based on these 3 pieces of information the KCA calculates the change that should be made to the kiln's rotary motor power in order to optimise the process and sends this information to the PCA. Based on this proposed change, the PCA and the KCA respectively calculate the appropriate speed of the precalciner fan and the fuel

rate for the burner in the kiln subprocess (they both calculate and exchange intermediate approximations before coming to a final value). These setpoint values are then fed to the application level where they are enacted on the physical process. This is an example of cooperation based on the sharing of intermediate and final results which produces more coherent control of the underlying process.

3.4 Control of a Particle Accelerator

(refs: Jennings *et al.*, 1993; Perriollat *et al.*, 1993)

This application study involves diagnosing faults in CERN's proton synchrotron (PS) accelerators. The PS complex is at the heart of CERN's accelerator and experimental facilities acting as an injector for the bigger accelerators (the Super Proton Synchrotron and the Large Electron Positron rings). For the PS accelerator complex about 10,000 control parameters have to be set correctly and many of them routinely change every 1.2 or 2.4 seconds to provide different types of beams to different users in a time sharing manner. The reasons for adopting a DAI approach to this application include: the desire to connect together two independently developed but related expert systems, the fact that the domain is simply too large to be embodied into a single monolithic system, the desire to preserve departmental boundaries, and the belief that future control systems are highly likely to be distributed in nature.

This study involved two diagnosis agents, both of which were originally conceived and implemented as standalone expert systems. BEDES diagnoses operational faults at the beam level. These faults occur, for example, if the intensity of the beam falls below a certain level or if the beam deviates considerably from its ideal trajectory. Such problems can be caused by the incorrect setting of a control parameter, a breakdown in a controller or an error in the control system itself. BEDES can diagnose the first two types of fault if the underlying control system is still working correctly. The other diagnosis agent, CODES, operates solely on the level of the accelerator's control system and can detect the final type of fault.

Faults in the control system often require BEDES and CODES to work together to determine the source of the fault - BEDES helps detect whether the problem is caused by wrong parameter settings or a breakdown in the controller, while CODES determines whether the fault is caused by the control system itself. In more detail, assume that BEDES has detected a fault and has initiated its diagnosis activities. As a first step it generates a list of hypotheses (assertions together with accompanying knowledge about how to prove them) to start working on. These hypotheses are communicated to CODES, to see whether they are manifestations of control system failings, where they form the start point of CODES' diagnosis process. Having generated this initial list, BEDES proceeds to perform a sequential detailed exploration of each hypothesis. The outcome of this exploration is that a particular hypothesis H is likely to be the cause of the fault (confirmed) or that H cannot be confirmed as the cause. This information is then sent to CODES where it is used to re-evaluate the priorities of its hypotheses (if H is highly suspected (confirmed) by BEDES then CODES should move it to the front

of its agenda or give the processing a higher priority because it indicates the region of the accelerator where the fault is most likely to lie; however if it is given a low likelihood of being the cause (not confirmed) then it should be moved to nearer the end of the agenda because BEDES can find no evidence to support the fact that the fault is in this area of the accelerator). This scenario illustrates how the sharing of partial and final results can be used to focus (or divert) the reasoning of an agent towards (away from) a promising (unpromising) area of the search space.

3.5 Control of a Flexible Robotic Cell

(refs: Oliveira and Ramos, 1993; Oliveira *et al.*, 1991)

The academic robotic testbed which has been developed at the University of Porto aims to be suited for assembly of units made up of different parts where millimetric precision is not required. Its main characteristic is to be flexible, being able to perform several different assembly tasks with the minimum amount of reprogramming. The testbed consists of a table top on which some objects have been placed in order to be manipulated by a 5.5 degrees of freedom robot arm. A DAI approach was adopted for this application because it provides a more natural representation of the inherently distributed real world domain and because breaking the problem and the problem solving strategy into smaller units makes the solution process simpler and easier to understand.

This application consists of 7 agents: a task level planner (TLP) which is a hierarchical and non-linear high level plan generator, an execution level planner and task executor (ELP-TE) which is responsible for executing the plan and table space management, a VISION agent which is responsible for object recognition, a LASER agent which is responsible for object recognition at the initial state, a MODELS agent which represents a database of the objects' features, a world description agent (WD) which extracts objects, relationships and constraints, and a USER agent.

In this testbed the user starts the system off by requesting the execution of an assembly task and giving a specification of the final configuration to the USER agent. The USER agent volunteers the final locations of the objects to all interested parties and requests that the ELP-TE agent execute the plan. Upon receipt of the request, ELP-TE determines what input it requires to execute the assembly task and sends subsequent requests to the appropriate agents (get initial object positions to the VISION and LASER agents, get object manipulation models to the MODELS agent, get geometric constraints to the WD agent and produce a high level plan to the TLP). To execute their respective tasks, most of the agents will require further information from their acquaintances - eg the TLP requires initial object locations and the geometric constraints and will therefore send requests onto VISION/ LASER and WD respectively. Eventually the relevant information will be provided and the appropriate tasks can be executed. When the TLP has all the information that it needs it will generate a high level symbolic plan and pass it back to the ELP-TE agent which translates it into a lower level set of actions. These actions are then down loaded to the PC controlling the robot which commands the execution of the corresponding task. Thus in this scenario

a number of agents are working together to generate a plan which can be executed by the ELP-TE agent - cooperation involves requesting tasks to be carried out by acquaintances, resolving conflicts that occur (eg discrepancies between the VISION and LASER agents), and volunteering relevant information to interested parties.

4. CONCLUSIONS

This paper has presented a brief resumee of the work undertaken in the ARCHON project. The agent and community architectures have been presented as have the applications to which this technology has been applied thus far. For the future, a number of new application studies are planned and a number of enhancements to the architecture have been proposed. These enhancements involve: (i) incorporating more sophisticated forms of cooperation into the Planning and Coordination Module (presently only asking acquaintances to execute particular skills (produce particular results) and the spontaneous volunteering of relevant information to interested agents are supported); (ii) improving the inter-agent coordination mechanisms (presently agents operate from a predominantly individualistic perspective and there is relatively little synchronisation of activities between agents); (iii) improving the ARCHON layer's ability to operate in real-time and deal with hard and soft deadlines (presently all the real-time capabilities of an ARCHON community reside within the intelligent systems); (iv) devising better means of presenting a community of cooperating agents to the user and improving the integration of the user as an active problem solver within the community; and (v) providing a more declarative and open means of sharing information between heterogeneous agents (the present mechanism based on federated schemas places a large burden on the application developer and limits the amount of run-time reasoning which the agents can perform).

ACKNOWLEDGEMENTS

The work described in this project has been carried out in the ESPRIT II project ARCHON (P-2256) whose partners are Atlas Elektronik, Framentec-Cognitech, Labein, Queen Mary and Westfield College, Iberdrola, EA Technology, Amber, Technical University of Athens, FWI University of Amsterdam, CAP Volmac, CERN and University of Porto. In writing this paper I am acting on behalf of the whole consortium by disseminating the project's results; I am 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), the PCM (Nick Jennings, Jeff Pople, Jochen Ehlers, Eugenio Oliveira), AIM (Frank Tuijnman, Hamideh Afsarmanesh, Giel Wiedijk), the HLCM (Claudia Roda, Jutta Mueller), the Agent Models (Nick Jennings) and the C++ implementation

(Rob Aarnts). Work in the applications was carried out by the following people electricity distribution (Andrew Cross, David Cockburn, Laszlo Varga), electricity transportation (Jose Corera, Inaki Laresgoiti, Juan Perez), particle accelerator control (Paul Skarek, Rob Aarnts, Laszlo Varga), cement factory control (George Stassinopoulos, Evangelos Lembesis, Robert King) and robotics (Eugenio Oliveira)

REFERENCES

Abel, E., Laresgoiti, I., Perez, J., and Corera, J., (1993) "Distributed Diagnosis in Electrical Networks" Proc. TOOLDIAG'93, Toulouse, France.

Cockburn, D., (1993) "Final Documentation of the CIDIM System" ARCHON Public Deliverable 980.

Cockburn, D., and Jennings, N. R., (1995), "ARCHON: A DAI System for Industrial Applications" in Foundations of DAI (eds. G. M. P. O' Hare & N. R. Jennings), Wiley Interscience (to appear).

Corera, J. Laresgoiti, I. Cockburn, D., and Cross, A. (1993), "A Cooperative Approach Towards the Solution of Complex Decision Problems in Energy Management and Electricity Networks", Proc. Int. Conf. on Electricity Distribution, Birmingham, UK, 4.19.1-4.19.6.

Jennings, N. R. (1994) "Cooperation in Industrial Multi-Agent Systems" Series in Computer Science - Vol 43, World Scientific Press (ISBN: 981-02-1652-1).

Jennings, N. R., and Pople, J. A., (1993) "Design and Implementation of ARCHON's Coordination Module" Proc. Workshop on Cooperating Knowledge Based Systems, Keele, UK, 61-82.

Jennings, N. R., Varga, L. Z., Aarnts, R., Fuchs, J., and Skarek, P. (1993), "Transforming Standalone Expert Systems into a Community of Cooperating Agents", Int. Journal of Engineering Applications of Artificial Intelligence 6 (4) 317-331.

Jennings, N. R., and Wittig, T., (1992) "ARCHON: Theory and Practice" in Distributed Artificial Intelligence: Theory and Praxis (eds. N. M. Avouris and L.Gasser), Kluwer Academic Press 179-195.

Lembessis, E., and Antonopoulos, G., (1993) "Report on the AMBER Application Case Study: High Level Control of a Kiln Complex in Cement Manufacturing" ARCHON Public Deliverable 1030.

Oliveira, E., Camacho, R., and Ramos, C., (1991) "A Multi-Agent Environment in Robotics" Robotica 9 (4) 431-440.

Oliveira, E., and Ramos, C., (1993) “Cooperation in the University of Porto Robotic Testbed” ARCHON Public Deliverable 1050.

Perriollat, F., Skarek, P., and Varga, L. Z., (1993) “Report on the CERN Application Study” ARCHON Public Deliverable 1060.

Roda, R., Jennings, N. R., and Mamdani, E. H., (1991) “The Impact of Heterogeneity on Cooperating Agents”, Proc. AAAI Workshop on Cooperation among Heterogeneous Intelligent Systems, Anaheim, USA

Stassinopoulos, G., and Lembesis, E. (1993), “Application of a Multi-Agent Cooperative Architecture to Process Control in the Cement Factory”, ARCHON Technical Report 43, Atlas Elektronik, Bremen, Germany.

Tuijnman, F., and Afsarmanesh, A., (1993), “Distributed Objects in a Federation of Autonomous Cooperating Agents” Proc. Int. Conf. on Intelligent and Cooperative Information Systems, Rotterdam, The Netherlands, pp 256-265.

Varga, L. Jennings, N. R., and Cockburn, D. (1994), “Integrating Intelligent Systems into a Cooperating Community for Electricity Distribution Management”, Expert Systems with Applications (1994) 7 (4)

Wittig, T. (Ed), (1992), “ARCHON: An Architecture For Multi-agent Systems”, Ellis Horwood Chichester.

NOTE: Copies of the ARCHON final deliverables can be obtained by writing to: Dr. Thies Wittig, Atlas Elektronik GmbH, Research & Development Projects, TEV2, SeebaldsbruckerHeerstrasse 235, D-2800 Bremen 44, Germany.