ARCHON: A DISTRIBUTED ARTIFICIAL INTELLIGENCE SYSTEM FOR INDUSTRIAL APPLICATIONS

D. Cockburn
EA Technology,
Capenhurst,
Chester,
CH1 6ES,
UK.
dac@eatl.co.uk

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

Abstract

ARCHON^{TM} (ARchitecture for Cooperative Heterogeneous ON-line systems) is Europe's largest project in the area of Distributed Artificial Intelligence (DAI). It has devised a general-purpose architecture, software framework and methodology which has been used to support the development of DAI systems in a number of 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 tackles 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 functional and implementation architectures. Thirdly, a preliminary methodology for developing ARCHON communities is sketched. Finally, the application of the software framework and the methodology is illustrated through the development of a DAI system in the electricity distribution domain.

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. The benefit of taking a holistic perspective is that the partial sub-systems 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 sub-components and a methodology which provides a means of structuring these interactions. ARCHON addresses both of these facets - providing a decentralised software framework for creating Distributed AI (DAI) systems in the domain of 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 sub-components 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. 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 (Varga *et al.*, 1994). 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.

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 outlines the ARCHON methodology, section four describes the electricity distribution application and section five shows how the software and the methodology were applied to this domain. Finally, section six presents the conclusions of this work and outlines some future plans.

2. The ARCHON Software

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 (see section 2.1). The design of the tasks invoked by the AL needs careful consideration if the AL is not to be swamped with vast amounts of low-level information which is irrelevant to its job of controlling cooperative problem solving. As a basic guideline, the tasks should be of sufficient granularity to be used as a decision point in the AL's reasoning about local control (further details are contained in sections three and four). This approach means that some tasks which appear indivisible to the AL may, in fact, be composed of several distinct processes in the IS.

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 (figure 14.1); 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.

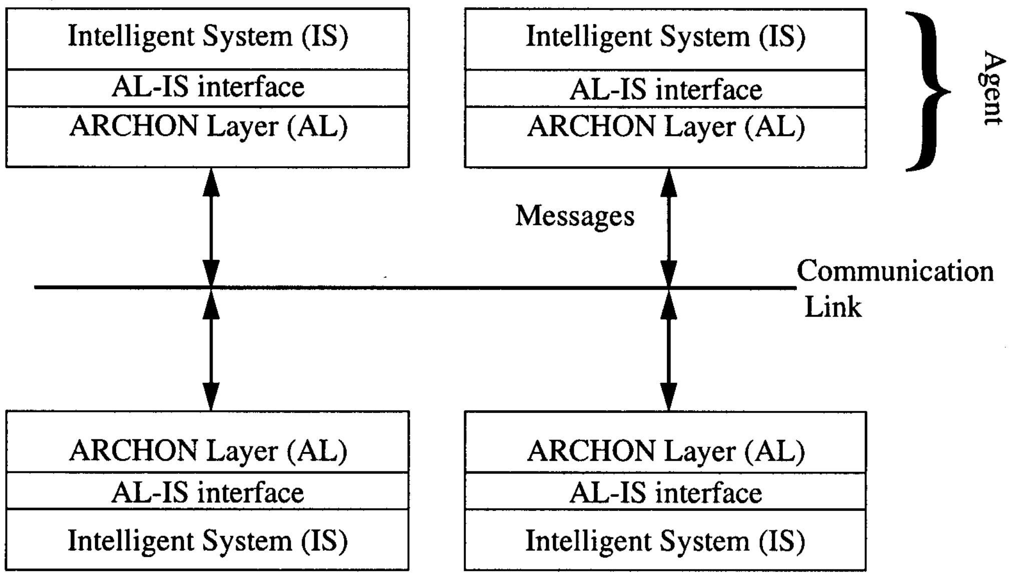


Figure 14.1: Structure of an ARCHON Community

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 other agents. The basic functional architecture highlights these four key aspects of **local control**, **decision making**, **agent modelling** and **inter-agent communication** (Figure 14.2). It was decided to embody this functionality in a modular and layered implementation architecture. This architecture contains four modules - **Monitor** (see section 2.1), **Planning and Coordination Module** (PCM) (see section 2.2), **Agent Information Management** (AIM) module (see section 2.3), and **High Level Communication Module** (see section 2.4) - each of which is responsible for one aspect of the functional architecture (Figure 14.3).

In terms of its hardware and software requirements, the ARCHON layer was originally implemented using LISP as a rapid prototyping language. Later it was ported to C++ in order to: improve speed, increase portability between different machines, and reduce the amount of computing resource needed by the AL. The C++ version runs under UNIX on Sun-4 architecture workstations (Sun-4s and SparcStations) and under Linux (a public domain version of UNIX) on i386 machines. There are also tools available which facilitate the construction of ARCHON communities (see section 4.6).

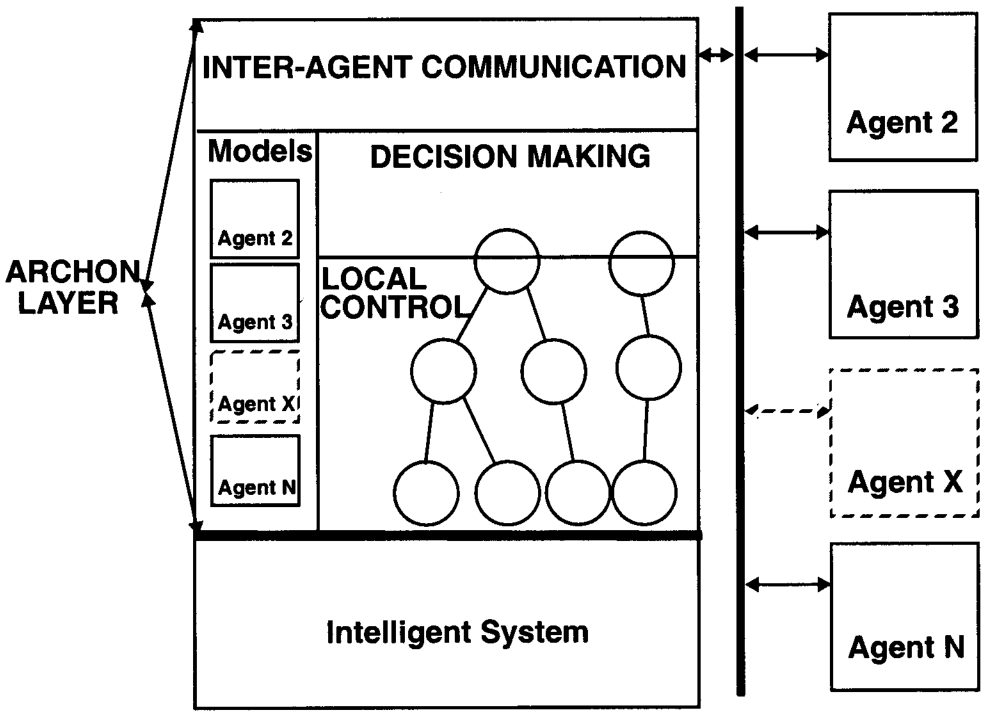


Figure 14.2: Functional View of an ARCHON Agent

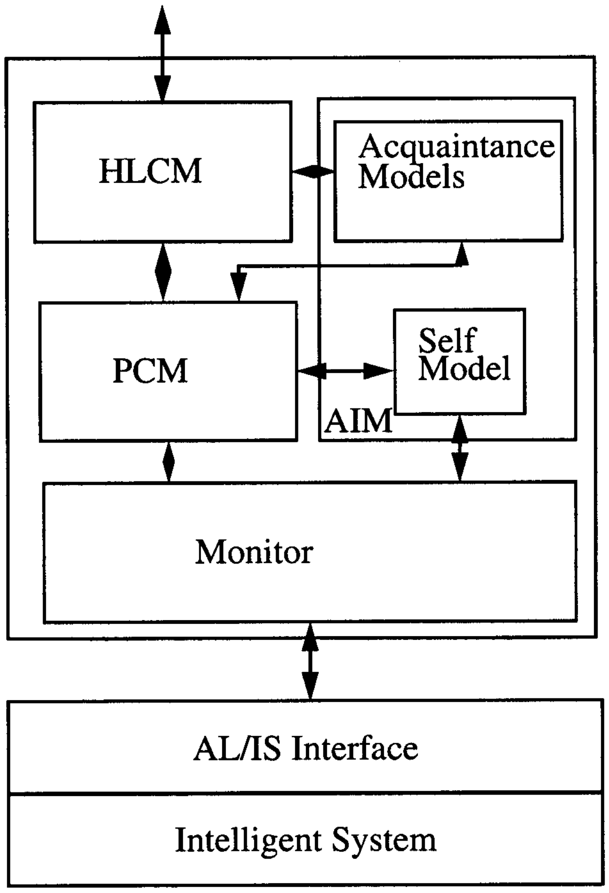


Figure 14.3: Implementation Architecture of an ARCHON Agent

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 14.4 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.

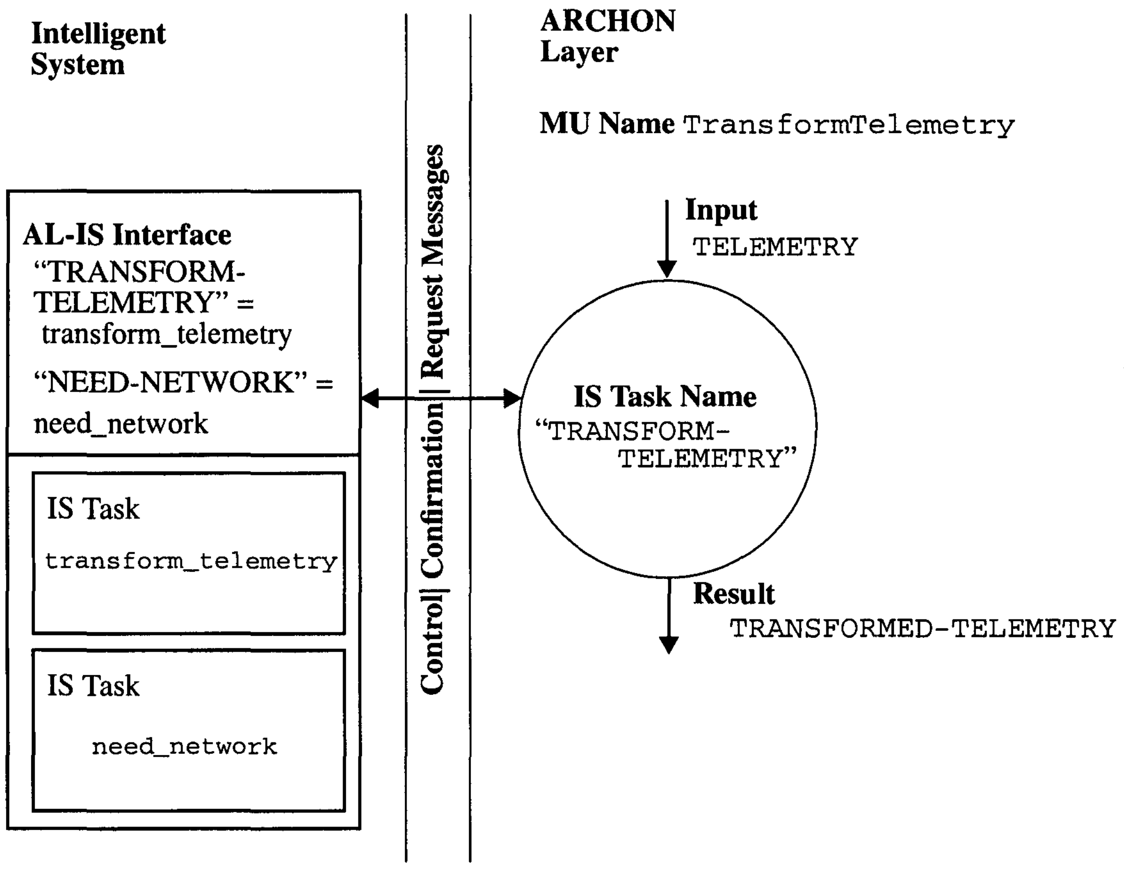


Figure 14.4: TransformTelemetry Monitoring Unit

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 (e.g. before a fault diagnosis process can be terminated any partial hypotheses it has asserted must be removed). 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.

In more detail, the message format between the AL and its IS is as follows:

```
<AL_IS_message> ::=
   (<message_type> <urgency> "<task_name" <tag>
              <argument_or_result>)
```

From the AL to the IS it is:

```
<message_type>::= <AL_IS_request> | <AL_IS_response>

<AL_IS_request> ::=
   start_is_task | kill_is_task | suspend_is_task |
   resume_is_task

<AL_IS_response> ::=
   requested_info | requested_info_not_available
```

and from the IS to the AL it is:

```
<message_type>::= <IS_AL_request> | <IS_AL_response>

<IS_AL_request> ::=
   information_request | intermediate_result

<AL_IS_response> ::=
   is_task_started| is_task_killed | is_task_suspended |
   is_task_resumed | is_task_completed
```

It can be seen that the AL can request to start, suspend, resume or kill any IS task and that the IS will provide an appropriate confirmation. The IS can also make a request to the AL for information, it can send intermediate results, and it can supply final results when a task completes. In this context, an intermediate result is one which is sent before the IS task has completed - this type of information is particularly useful for a monitoring task where it would be inefficient and disruptive to the IS's normal flow of control to produce a function which completed and returned a result.

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
- Be used to return intermediate results before a plan has completed

A sample plan which ensures that the electricity network is loaded is shown in figure 14.5. This plan is activated in a fault diagnosis agent as part of its diagnosis routine. Firstly, the `TELEMETRY` is transformed into a suitable format by the `TransformTelemetry` MU. Secondly, the MU `NeedNetwork` 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.

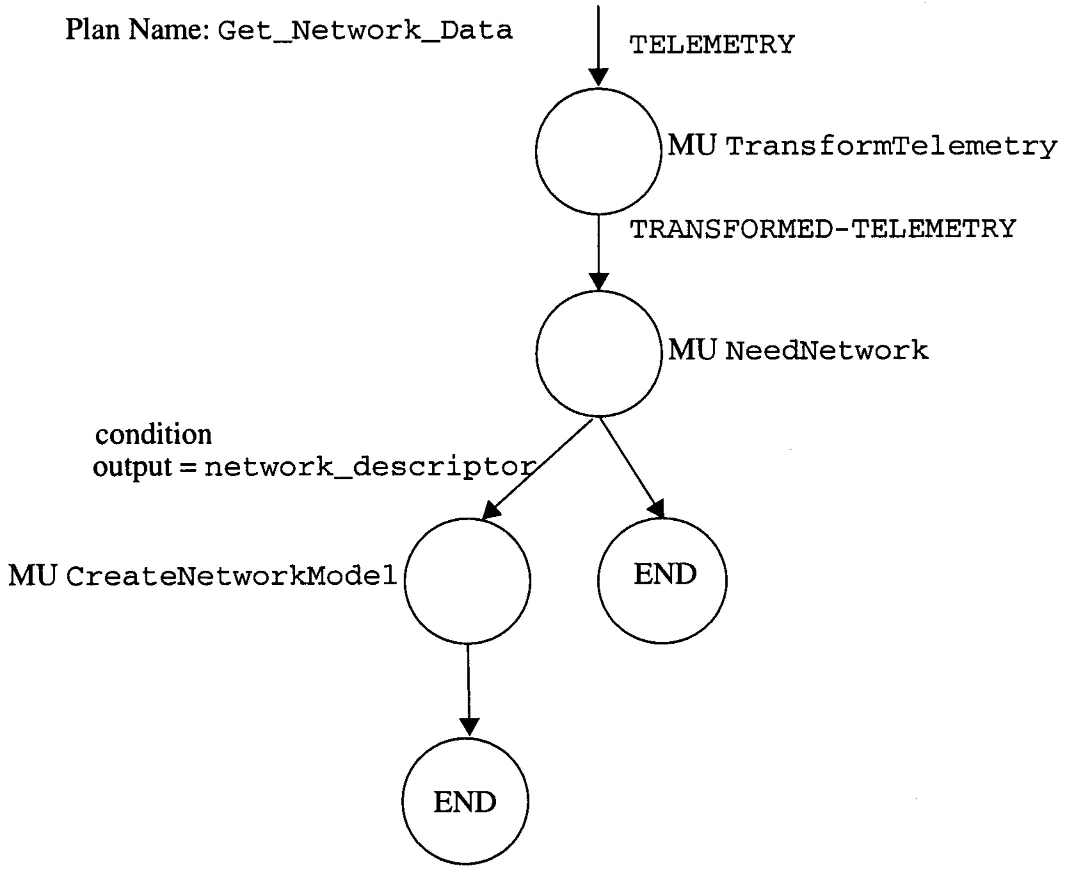


Figure 14.5: Get_Network_Data Plan

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[1]. There are two types of behaviour: those that are visible to the PCM (and the other AL components) and those that are purely internal to the Monitor (e.g. `DiagnoseFault` in

1. The reason for allowing behaviours to be linked is that it permits greater modularity: a plan can be written to perform one task and then it can be incorporated into several different behaviours.

figure 14.6). The former type are called **skills** (e.g. `DealWithTelemetry` in figure 14.6) and they may be triggered by data arriving from other agents or by direct requests from other agents.

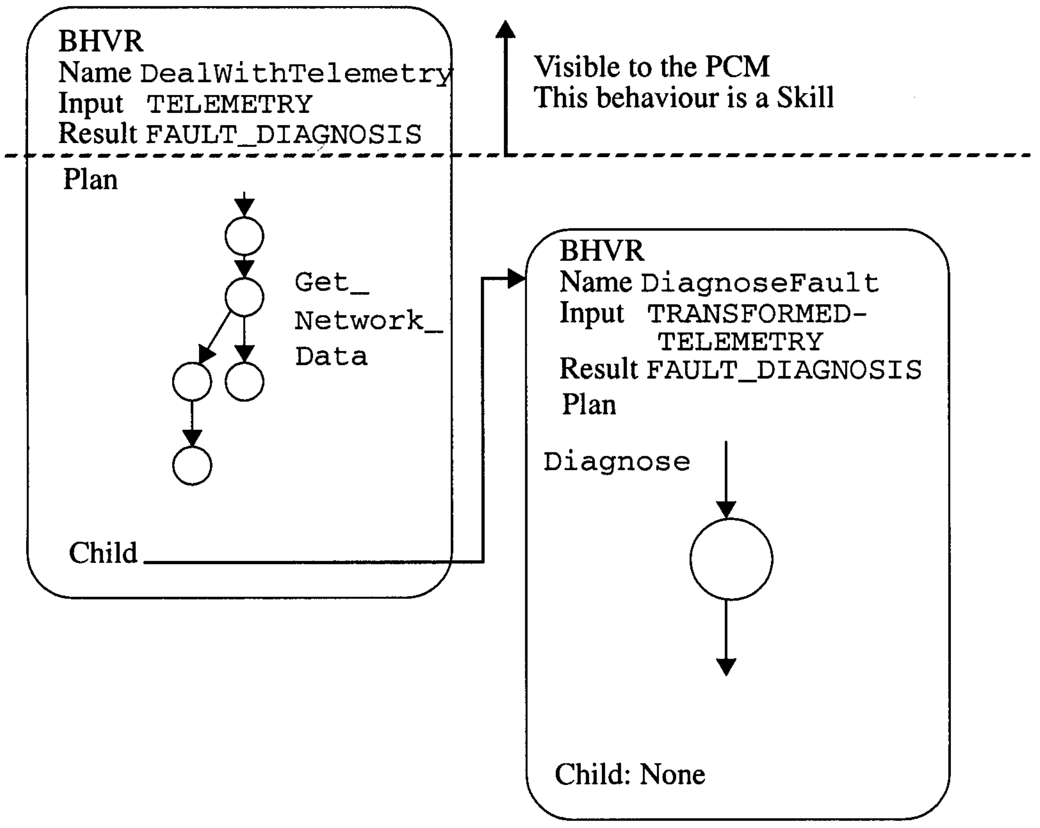


Figure 14.6: DealWithTelemetry skill with DiagnoseFault child behaviour

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 include: 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 (Jennings, 1992). 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, but not to the IA or HVDA 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. The ARCHON Methodology

As already mentioned, the design of an ARCHON system takes a mixed top down and bottom up approach. Taking a top down approach means the designer must look at the overall problem that the system is being asked to solve and identify the corresponding system-level goals. Next, the tasks necessary to achieve these goals need to be identified and suitably decomposed. Finally, the data flows between the tasks need to be determined. However this classical design perspective is not the full story in the type of applications with which we were faced - in many cases the system designer wanted to utilise existing ISs to provide some of the functionality. There are sound reasons for doing this: saving development time, familiarity with existing systems - both by the designer and the eventual end user - and proven technology.

For this reason, ARCHON has also been designed to allow pre-existing (legacy) systems to be integrated. However the use of such systems constrains the task decomposition and distribution processes (if these systems are to be used in a similar way to their original standalone counterparts). As an example, using an existing IS to provide a particular set of functionality typically implies that all the tasks will be co-resident in the same agent, whereas in a totally unconstrained distribution they may have been dispersed into multiple agents. It is at this pragmatic level where the bottom up considerations are of primary concern. The designer must examine the capabilities of the existing systems (in terms of their goals and tasks) and then compare these tasks with those suggested by the top down approach. Typically there will be a mismatch, since: the pre-existing systems do not provide all of the necessary functionality, some pre-existing system functionality is not required in the cooperating system, and some functionality is duplicated

Having performed this analysis, the designer must decide which of the pre-existing system's functionality should be exploited, which functionality necessitates the creation of new ISs, and which functionality can be supported by the AL. If new systems are to be created then it is important that they too have a clear functional role (rather than simply being a collection of tasks which do not fit in elsewhere). Some adaptation of existing systems may be required in order to ensure that the necessary tasks can be invoked by the AL (see Jennings *et al.* (1993) for a detailed analysis on a real-world example). Typically this process aims to improve the structure of the IS; for example, a standalone system may print results to the screen as they are produced and never build up an internal representation. A possibly useful reconceptualisation could be to develop a data structure to contain the full result and then return the complete answer from a suitable accesssor function.

It may be the case that there are a number of pre-existing ISs from the same domain that can perform the same task, but that the systems themselves are at different stages of development. For example, one IS may have been used in the real world and have an existing connection to it to obtain data. Another may require the same data but have only a simple interface (e.g. it is still in a prototype stage and reads data from a file). By integrating these two systems, it is possible to use the better developed interface for both systems and thus provide a significant improvement to the prototypical IS.

An important aspect of the bottom up methodology includes a thorough analysis of the data to be exchanged - this is important because the semantics of the data, as well as its syntax, need to be uncovered before meaningful interchange can take place (e.g. it is possible for two similarly named data types to have totally different meanings). This analysis is especially crucial when the cooperating community involves components which have been developed at different times and with which the designer has different levels of familiarity. A common source of such confusion occurs when data from different sources is not entirely compatible in all aspects. As an example consider the notion of 'time'. There are many reasons why 'time' in one IS differs from 'time' in another: it may be a matter of format (e.g. the time is in seconds rather than minutes from a given base time - the base time might differ too!), or, more importantly, although seemingly related they may measure different attributes (e.g. one could be a measure of CPU time used and the other of system time). This point is worth stressing because when a system is run in stand alone mode the absolute meaning of the data may not be as important to the user as a comparison between different outputs (e.g. he may have forgotten what the units are but knows what a good reading is and how to compare readings). Problems arise when two systems are integrated and it is thought that two (or more) data types can be easily interchanged because they appear similar.

ARCHON uses a standard data representation mechanism which allows the AL to make decisions based on the contents of the IS data. For example, the interests expressed in the acquaintance models can be dependant on the value of certain attributes of the data (e.g. agents diagnosing faults on electricity networks are interested in telemetry messages reporting the change of state of switches - see section 2.3), as might the triggers which activate behaviours. However this is not to say that the AL representation is imposed on the IS, indeed the IS can represent data in any way that it chooses, but rather that it can be used as a means of transferring data. New ISs can be designed to use this format and existing ones can be adapted to understand it. If there really is a necessity for an IS to represent data in a totally different way then this can also be accommodated. For example, data can be left in its original format in the message - this means the AL cannot make decisions based on that part of the data - or a file name can be sent where the data can be accessed. The latter relies on a common file system but is very useful for large amounts of structured data which the AL need not understand.

4. Using ARCHON to Manage an Electricity Distribution and Supply Network

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 (Cockburn *et al.*, 1992; Varga *et al.*, 1994)
- Electricity transmission and distribution (Corera *et al.*, 1993)
- Control of a cement kiln complex (Stassinopoulos and Lembessis, 1993)
- Control of a particle accelerator (Jennings *et al.*, 1993)
- Control of a robotics application (Oliveira *et al.*, 1991)

In this paper we concentrate on the first of these applications. The DAI system which has been developed is called **CIDIM** (Cooperating Intelligent systems for DIstribution Management systems) and its aim is to help control engineers manage electricity distribution and supply networks.

4.1 The CIDIM Application

CIDIM is being developed as an aid to the control engineer who is responsible for ensuring the continuity of electricity supply to customers. The main jobs to be carried out include: planning and carrying out maintenance work safely and in coordination with the field engineer, identifying faults on the network, and taking action to restore supply should this be necessary. The electricity network control system allows remote operation of circuit breakers and reports, via telemetry, automatic switching operations in response to faults, alarms and load readings. The control system covers the high voltage network and part of the low voltage network, but for much of the low voltage network switching for maintenance purposes is done manually by the field engineer in radio contact with the control engineer. Due to the lack of telemetered protection equipment, customer telephone calls reporting loss of supply play an important role in decision making about the low voltage network. The final important source of information used by the control engineer concerns lightning strikes - these may be the cause of a fault and so indicate a good starting point for the field engineer to look for damaged equipment.

CIDIM assists the control engineer by: (i) automatically providing a comprehensive range of services such as fault diagnosis, lightning detection, user driven restoration planning and automatic rechecking of restoration plans; (ii) automatically collating much of the information which is currently collected manually by reference to stand-alone systems. ARCHON also permits information from conventional knowledge sources, such as a database or the telemetry system, to be shared by more than one agent within the community (thus increasing the degree of consistency).

4.2 Tailoring the ARCHON Methodology to CIDIM

The top down aspect of the ARCHON methodology for the CIDIM application is obtained by examining the role of the control engineer. The main aspects of this job are as follows:

- Ensure continuity of supply to customers
- Supervise maintenance on the network
- Restore power after faults

The tasks which are performed in order to meet the above goals are:

- Create safe switching plans for maintenance and repair
- Diagnose faults on the network

These are the goals and tasks at the highest level. These activities were then broken down further; descriptions were made of: the individual tasks (e.g. fault diagnosis and restoration planning), the data requirements for these tasks (e.g. telemetry, lightning data and network data), and the inter-relations between the tasks and the data (e.g. fault diagnosis requires telemetry and network data).

The bottom up aspect of the methodology for CIDIM was obtained by examining the functionality of the pre-existing and standalone systems which were currently used in the Control Room. These systems were:

- High Voltage Expert System (HVES) (Cockburn *et al.*, 1991a; Cockburn, 1992)

  Diagnoses the location and type of high voltage fault on the electricity network. To do this it uses telemetered information from the network protection system (which automatically operates circuit breakers in order to first isolate the faulty section from the rest of the network and then, if possible, restore power). Originally, the HVES had its own network representation and a separate process for accepting telemetry. It was designed so that it could still work if telemetry messages were missing although sometimes it was unable to discriminate between competing alternatives if it did not have sufficient information.

- Switching Schedule Production Assistant (SSPA) (Cross *et al.*, 1992)

  When there is a permanent fault, or a need for maintenance work, a safe switching plan needs to be created. This ensures that the area to be worked on is isolated from the rest of the network and that it is safe for the field engineers to start their work. Originally, the SSPA had its own network representation.

- Weather Watch System (Scott, 1988; Lees, 1992)

  The location of lightning strikes can be useful supplementary data to the control engineer. It can help in fault location on overhead lines - on a long line it is best to start to look for damage near to a lightning strike - and can also warn the field engineer that it is unsafe to work in a particular location.

4.3 Applying the ARCHON Methodology to CIDIM: High Level System Design

By comparing the requirements from the top down analysis with the functionality of the existing systems, it can be seen that:

- There is no assistance for low voltage fault detection. Telemetry data is only present at the point where the high and low voltage networks join and it alone cannot locate the fault or identify its type. However information from customers' telephone calls which report their loss of supply can be used.

- Sometimes the control engineer will want to operate automatic circuit breakers from the control room in an attempt to restore power by an alternative route. In such cases, a check to see if these operations are safe will help minimise the chance of costly mistakes.

- The overall security of the network is not considered. Rather than just waiting for, and reacting to, faults the control engineer could proactively switch out overloaded lines and reroute power.

In order to provide the missing functionality it was decided that the following additional ISs were required to respectively deal with each of the above points: (i) low voltage expert system (LVES); (ii) switch checking system (SCS); and (iii) security analysis system. Note that the last two systems have yet to be fully integrated into the CIDIM cooperating community.

With the development of these new ISs, the overall system provides most of the functionality required by the control engineer. As several of the systems make use of telemetry (HVES, LVES) and require network data (HVES, LVES, SSPA, SCS), it was decided to create dedicated agents to provide these services - respectively, the telemetry agent (TA) and the information agent (IA). As the HVES already had a program running as a separate process which accepted telemetry from the network and translated it into a standard format, it was decided to make this program into the TA. The TA could then provide this service, and a standard format, to the other community members. In the UK, different Regional Electricity Companies (RECs) have different telemetry formats and so moving CIDIM to another REC would simply require changes to the IS of the TA program (rather than to all the agents using telemetry). The IA holds the network database for all agents; again although each REC has a different database, and a different structure within that database, this approach only requires changes to the IS of the IA when CIDIM is ported to a different company.

It was decided to provide a common user interface to CIDIM because one of its major functions is to collate and present integrated information from different sources (e.g. when there is relevant information about lightning strikes it needs to be presented in conjunction with the output of the fault diagnosis activity if it is to have the maximum impact). The common interface also means that the control engineer can view CIDIM as a single system and need not be aware of the source of the results (e.g. fault reports from the HVES and the LVES are simply reported as faults - if they are displayed differently it is because they relate to different voltage levels and not because they are produced by different agents).

Both the HVES and the SSPA already had graphical displays of the electricity network, but as the SSPA's interface was more sophisticated it was used as the basis of the system's common interface (named the Advisor Agent). For reasons of familiarity, it is still possible to use the existing interfaces for some of the systems; for example, the SSPA is user driven and when creating a switching schedule it is easier and more efficient to use the existing display (this is possible because there is no need for interaction with the rest of the CIDIM system when creating a switching schedule). The CIDIM system as it now stands is shown in Figure 14.7.

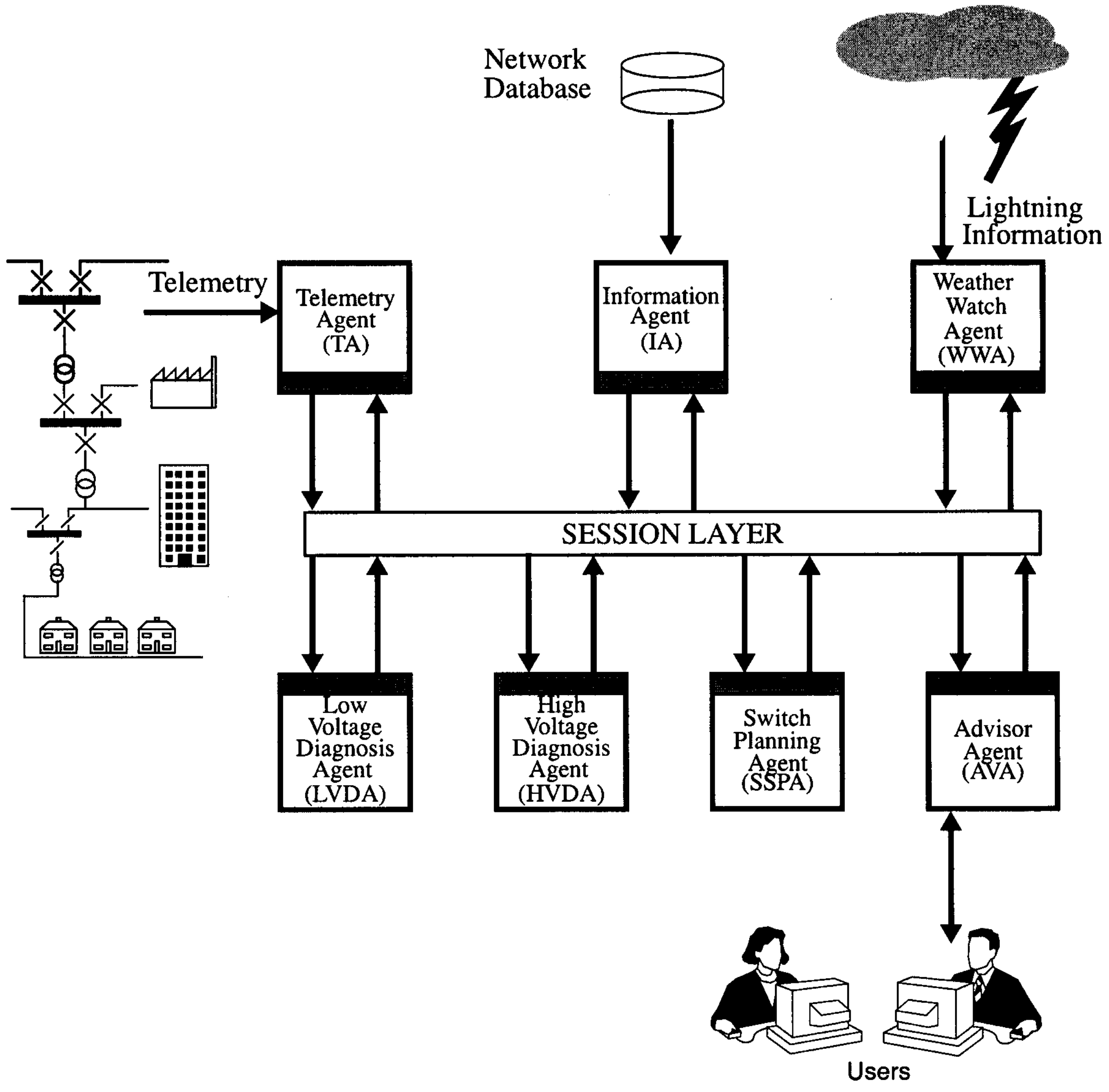


Figure 14.7: The CIDIM Community

Integrating these different sub-systems in CIDIM allows the agents to interact, this, in turn, gives the following benefits over and above their standalone counterparts:

(i) Automatic look up and cross referencing of lightning data when there is a fault.

(ii) Display of all relevant results in an integrated manner.

(iii) Interactions between the high and low voltage diagnosis agents and the TA can resolve conflicts in high voltage diagnosis when telemetry is missing.

(iv) Notification of faults in areas where work is planned.

The first two points are dealt with earlier in this section, hence we concentrate on the remaining cases here. Point (iii) illustrates how an integrated system is able to provide the control engineer with better quality information. The HVES can diagnose faults when some telemetry information is missing, however in such cases its diagnosis can be less precise about the nature (permanent/transient) or location of the fault. In CIDIM these shortcomings can be minimised in two ways. Firstly, the fact that a permanent fault on the high voltage network can lead to a loss of supply on the low voltage network, since the former feeds the latter, can be exploited. In such a case the high voltage diagnosis agent can ask the low voltage agent if the low voltage network has been affected. The second improvement is that the high voltage agent can ask the TA whether the substation from which the telemetry is missing is working. Replies from each of these agents will assist the high voltage agent in deciding on the nature and location of the fault.

Point (iv) is one illustration of how information passing between agents can trigger useful actions. When the high voltage agent produces a diagnosis it will send it to the SSPA which will then recheck any preplanned restoration plans on the updated state of the network. If this analysis results in the conclusion that the plans need to be redone, because of the effect of the fault, then the user is automatically informed.

4.4 Agent Design

The key design decision within an individual agent concerns the interaction between the AL and its underlying IS. Assuming the agent's high-level role has been clearly defined (see section 4.3), the functionality which is of use to other agents must be identified, as must the appropriately abstract logical grouping of the local functionality.

To illustrate these concepts in a concrete manner, the design of CIDIM's low voltage diagnosis agent (LVDA) will be expanded upon in detail. This instantiation is viewed from a bottom up perspective - i.e. the level at which the IS and the AL interact is determined, then the MUs, plans, behaviours and skills are determined, and, finally, the self and acquaintance models are populated. The LVES diagnoses faults using telemetry and telephone calls - a diagnosis can be triggered by either type of information and the outcome may rely on more than one piece of telemetry or more than one telephone call. The diagnosis process is rule based and relies on the incoming data to update its internal representations of the hypotheses. A firm diagnosis is only made when there is a sufficient degree of certainty that all the relevant indicators have arrived. The time the system should wait for data to arrive is dependent on the nature of the fault and is reassessed each time new data arrives. Before the diagnosis can commence the relevant portion of the electricity network needs to be loaded.

4.4.1 IS Tasks and Monitoring Units

ARCHON is designed to have a relatively coarse granularity of interaction between the AL and its IS - indeed it is inefficient to introduce unnecessary interactions. As mentioned in section 2.1, the IS must be conceptualised as a number of functional units. Even if the IS is rule based, and works by side effect, this is still possible as the following example from the LVDA illustrates.

One of the activities performed by the LVES is to diagnose faults using telemetry information as an input. As shown by figure 14.8, this activity involves a complex series of actions within the IS: the first thing to check is whether the telemetry relates to an existing hypothesis or whether a new hypothesis is needed. In the latter case, the IS must set a timer to indicate when it can report its final diagnosis - this depends on the information received and an estimation of the time by which the remaining necessary information will arrive. In the former case, there will already be an active behaviour concerned with this hypothesis and so a timer will already have been set, but the IS may need to update the hypothesis and possibly reset the timer (e.g. if the new data represents the final piece of expected information then the timer can be set to zero).

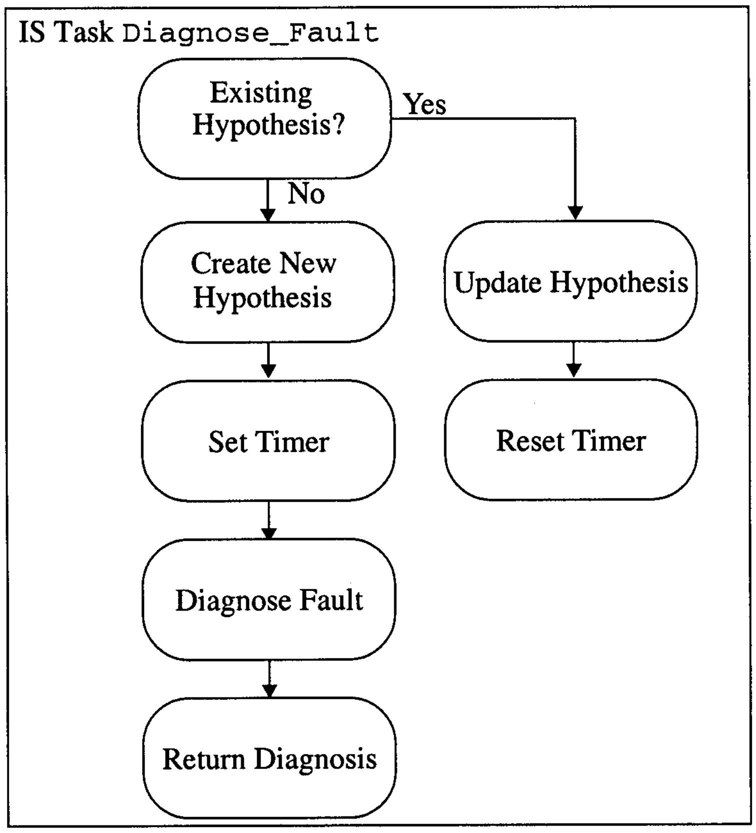


Figure 14.8 The IS Task Diagnose_Fault

Despite this complexity, it was decided to encode this process in a single IS task because there is no further need for interaction with the AL before the task completes. The corresponding MU is represented in the AL as follows:

```
MU DiagnoseFault             ;; MU Name
"Diagnose_Fault"             ;; IS Task Name (see fig. 14.8)
(TRANSFORMED-TELEMETRY)      ;; mandatory input(see fig. 14.6)
(FAULT_DIAGNOSIS)            ;; result (see fig. 14.6)
```

4.4.2 Plans

The `Diagnose` plan which performs the diagnosis is extremely simple and just involves calling the `DiagnoseFault` MU (see figure 14.6). In contrast, the `Get_Network_Data` plan which loads the relevant network before the diagnosis is made is more complex (see Figure 14.5). After the telemetry has been suitably transformed, the MU NeedNetwork checks whether the network from which the telemetry originated is represented internally in the IS. If it is not then the IS extracts a network descriptor, by which the relevant portion of the network can be accessed, from the telemetry message and returns it to the Monitor as the result of the MU; if the network is already available then no result is returned. Next there is a decision point: the Monitor checks if a network descriptor was returned, if not then the network is already available and the plan ends; however if a network descriptor was returned then a request is made to the PCM for the network (since it cannot be produced locally). The PCM checks its acquaintance models, sees that the IA can provide the network (see section 4.4.4), and so sends it a request for assistance. On return of the network, an appropriate model is established by the CreateNetworkModel MU.

4.4.3 Behaviours and Skills

To allow for easier testing and expandability two behaviours were created, DealWithTelemetry and DiagnoseFault (see Figure 14.6), to control the diagnosis process with respect to telemetered information. DealWithTelemetry is the skill visible to the rest of the ARCHON Layer, being a child behaviour DiagnoseFault is internal to the LVDA's Monitor. The trigger which fires the DealWithTelemetry skill can be expressed in the following manner:

```
TRIGGER DealWithTelemetry-Trigger
(TELEMETRY)              ;; Arrival of this info is the trigger
[                        ;; Actions associated with trigger
  EXECUTE(DealWithTelemetry);
]
```

4.4.4 Self and Acquaintance Models

The self model of the LVDA contains a description of the skills which can be executed locally. As well as DealWithTelemetry (shown below), there is DealWithTelephoneCalls (which controls fault diagnosis using customer reports of loss of supply), and ResolveHV_Conflict (which the high voltage agent will request in order to determine whether or not a high voltage fault has affected the low voltage network - refer to the discussion in section 4.3).

```
Name: DealWithTelemetry
Trigger: DealWithTelemetry-Trigger    ; see section 4.4.3
Inputs: ((TELEMETRY :mandatory))
Results: (FAULT_DIAGNOSIS)
Plan Name: Get_Network_Data
Children: (DiagnoseFault)
```

In order to perform its designated role in CIDIM, the LVDA only needs to be aware of two other agents: the information agent and the advisor agent. For the IA it represents the fact that it can perform the skill GetNetworkSubstation (which needs to be executed to provide the network required by the CreateNetworkModel MU of the Get_Network_Data plan):

```
Name: GetNetworkSubstation
Trigger: ()
Inputs (network_descriptor)
Results (NETWORK)
```

For the advisor agent, the acquaintance model indicates that it is interested in the outcome of the diagnosis (to show to the control engineer):

```
INTEREST-DESCRIPTOR
   INFORMATION-NAME: FAULT_DIAGNOSIS
   INFORMATION-CONDITION:
      [
         ("AVA", TRUE);
      ]
```

It can be seen that although the LVDA does have interactions with the TA and the high voltage agent it does not model them. This is because the models are constructed on a need-to-know basis: the LVDA simply receives telemetry from the TA and so does not need to know about its interests and skills; the high voltage agent makes requests of the LVDA, but not vice versa, so the high voltage agent models the LVDA but not the other way around. Further details of these models can be found in Cockburn *et al.* (1991b) and Cockburn (1993).

4.5 An Example Cooperative Scenario

Below a simple scenario showing how the LVDA interacts with the telemetry agent (TA), the information agent (IA), and the advisor agent (AVA) to diagnose and report a fault on the low voltage network is described. The names of MUs, plans, and behaviours are those used in earlier sections.

The TA is continually receiving telemetry from the electricity network. This is transformed into the standard ARCHON format by its IS and is made available to the PCM as an intermediate result. The `TELEMETRY` is checked against its acquaintance models, by the PCM, and is sent to the LVDA if its operation is autoreclosure (see section 2.3).

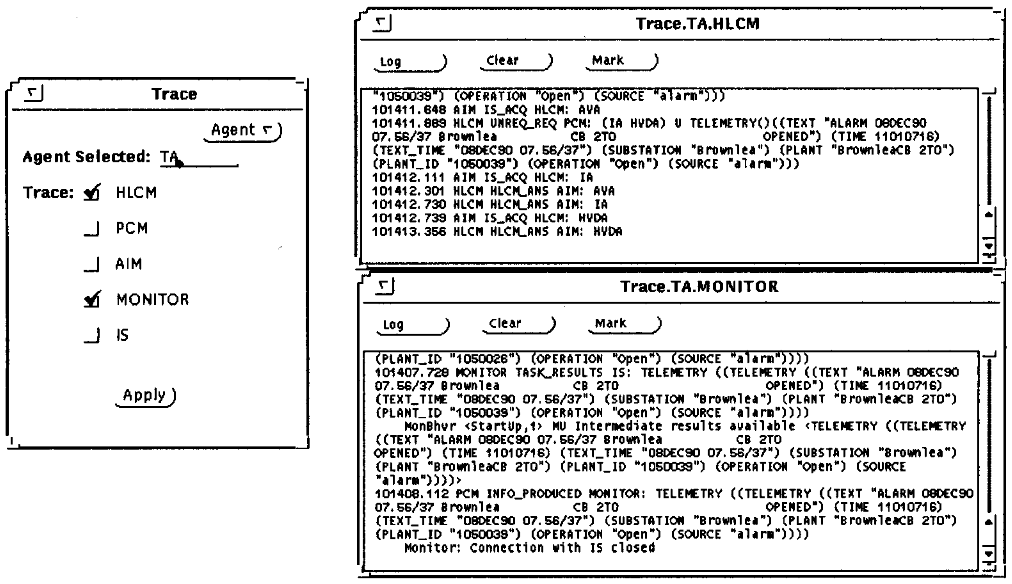
On arriving at the LVDA, the PCM checks to see if any skills will be triggered by the incoming data. In this case, the skill `DealWithTelemetry` is triggered (see section 4.4.3) and the `TELEMETRY` data is added to its context. Control then passes from the PCM to the Monitor in order to execute the associated plan (`Get_Network_Data` - see figure 14.5). The first MU, `TransformTelemetry`, is invoked - its input, `TELEMETRY`, is extracted from the context of the behaviour. The Monitor creates a message containing the name of the IS task (`TRANSFORM-TELEMETRY` - figure 14.4) and the `TELEMETRY` data and sends this to the IS via the interface. The task is performed and the result, `TRANSFORMED-TELEMETRY`, is added to the context of the behaviour. The next MU in the plan, `NeedNetwork`, is then activated - this checks if the network has been loaded for the `TRANSFORMED-TELEMETRY`. Next the plan forks and so, by convention, the leftmost branch is tried first. This branch contains a condition checking on the output of the previous MU. If a network descriptor has been produced then the condition is satisfied and the `CreateNetworkModel` MU is activated. Assuming this to be the case, `CreateNetworkModel` has `NETWORK` as an input. The Monitor checks to see if this is available within the behaviour's context: it is not, so a request is made to the PCM for the data. The PCM first checks, by referring to the self model, to see if the LVDA can produce the data itself. As it cannot, a check on the acquaintance models is made. It is seen that the IA can provide the data using the GetNetworkSubstation skill and that this skill requires the data network_descriptor to be sent from the behaviour's context (see section 4.4.4). The request is passed onto the HLCM where the IA's address is found and the message is actually delivered. The request is dealt with by the IA (with reference to its self model) and the appropriate portion of the network is returned. The LVDA's PCM then makes this result available to the behaviour from which the request originated. The CreateNetworkModel MU is then activated and upon its completion the plan ends.

As the DealWithTelemetry behaviour has finished, its child behaviour DiagnoseFault is called. The plan Diagnose is started and the MU DiagnoseFault is activated with input TRANSFORMED-TELEMETRY (see section 4.4.4). If this produces a new fault diagnosis then it will be some time before the MU returns a result. In the meantime, new telemetry and telephone calls will arrive, some of which will start new diagnoses and some of which will add to the one started by the original behaviour. When the DiagnoseFault MU completes the behaviour DiagnoseFault terminates, as does its parent DealWithTelemetry. On completion of the behaviour, the result (FAULT_DIAGNOSIS) is made available to the PCM. Note that the results of the other MUs (e.g. TRANSFORMED-TELEMETRY) are not specified as a result of the behaviour and so are not passed on. The PCM checks the self and acquaintance models to determine whether the diagnosis can be used locally (it cannot) or by another agent (it can - the AVA is interested (see section 4.4.4) and so it is sent the result).

4.6 System Debugging

Debugging in a multi-agent system is an even more complex and time consuming process than it is for standalone systems since more than one entity is being tested at the same time. In CIDIM, this process was exacerbated by the fact that some of the ISs were still under development when the cooperating community was being designed. To reduce the scope for errors it was decided that dummy ISs would be used for the preliminary testing phase (dummy ISs are stubs which return a result without computation). These stubs were then incrementally replaced with the real programs and a further round of testing was undertaken. The benefits of using this approach are as follows: the possibility of errors is reduced, the IS gives a known response, and the overall time for system testing is decreased. Also this mode of operation is a necessity when an agent is created from the first IS - since checks must be made to ensure that the agent responds properly to requests and volunteered information which arrive from its acquaintances.

An important feature of the ARCHON development environment which made this testing process considerably easier was the ability to trace each of the agent's modules in terms of the flow of messages between them. Figure 14.9 shows telemetry being received from the TA's IS as an intermediate result of the behaviour StartUp. The HLCM trace window shows this result being sent to the IA, the AVA and the HVDA as unrequested information because of the contents of their respective acquaintance models.



```
"1050039") (OPERATION "Open") (SOURCE "alarm")))
101411.648 AIM IS_ACQ HLCM: AVA
101411.889 HLCM UNREQ_REQ PCM: (IA HVDA) U TELEMETRY()((TEXT "ALARM 08DEC90
07.56/37 Brownlea           CB 2TO                        OPENED") (TIME 11010716)
(TEXT_TIME "08DEC90 07.56/37") (SUBSTATION "Brownlea") (PLANT "BrownleaCB 2TO")
(PLANT_ID "1050039") (OPERATION "Open") (SOURCE "alarm")))
101412.111 AIM IS_ACQ HLCM: IA
101412.301 HLCM HLCM_ANS AIM: AVA
101412.730 HLCM HLCM_ANS AIM: IA
101412.739 AIM IS_ACQ HLCM: HVDA
101413.356 HLCM HLCM_ANS AIM: HVDA
```

```
(PLANT_ID "1050026") (OPERATION "Open") (SOURCE "alarm"))))
101407.728 MONITOR TASK_RESULTS IS: TELEMETRY ((TELEMETRY ((TEXT "ALARM 08DEC90
07.56/37 Brownlea          CB 2TO                        OPENED") (TIME 11010716)
(TEXT_TIME "08DEC90 07.56/37") (SUBSTATION "Brownlea") (PLANT "BrownleaCB 2TO")
(PLANT_ID "1050039") (OPERATION "Open") (SOURCE "alarm"))))
    MonBhvr <StartUp,1> MU Intermediate results available <TELEMETRY ((TELEMETRY
((TEXT "ALARM 08DEC90 07.56/37 Brownlea          CB 2TO
OPENED") (TIME 11010716) (TEXT_TIME "08DEC90 07.56/37") (SUBSTATION "Brownlea")
(PLANT "BrownleaCB 2TO") (PLANT_ID "1050039") (OPERATION "Open") (SOURCE
"alarm"))))>
101408.112 PCM INFO_PRODUCED MONITOR: TELEMETRY ((TELEMETRY ((TEXT "ALARM 08DEC90
07.56/37 Brownlea          CB 2TO                        OPENED") (TIME 11010716)
(TEXT_TIME "08DEC90 07.56/37") (SUBSTATION "Brownlea") (PLANT "BrownleaCB 2TO")
(PLANT_ID "1050039") (OPERATION "Open") (SOURCE "alarm"))))
    Monitor: Connection with IS closed
```

Figure 14.9: Tracing Facilities of the ARCHON Toolbox

5. Conclusions and Future Plans

This paper has outlined the main features of the ARCHON approach to building real-world DAI applications in the industrial supervision and control domain. By necessity, some aspects of this description are lacking in detail but in such cases we have made reference to more comprehensive literature. Although the ARCHON system has been primarily designed based on our experiences of the supervision and control domain, it is felt that the architecture and the methodology will generalise, in a relatively seamless manner, to new application areas. Indeed this possibility is being actively pursued by a number of organisations!

In its present form, the CIDIM application consists of seven agents - the telemetry agent, the information agent, the weather watch agent, the low voltage diagnosis agent, the high voltage diagnosis agent, the switch planning agent, and the advisor agent. The community was tested by providing it with data gathered from a number of actual disturbance episodes. During these trials the agents cooperated with one another, in the manner described, to achieve the goals outlined in this paper. For the future, two avenues of exploitation are being pursued. Firstly, the possibility of running the CIDIM system in field trials is being investigated. Secondly, the ARCHON technology, and some of CIDIM's intelligent systems, are being offered to regional electricity companies as a way of improving the level of integration in their control rooms.

ACKNOWLEDGMENTS

The work described in this paper has been carried out in the ESPRIT II project ARCHON (P2256) 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. The concepts reported in this paper are the result of the consortium's efforts; however, we would particularly like to acknowledge the help provided by László Varga on this application.

REFERENCES

Cockburn, D. McDonald, J. Burt, G. Brailsford, J. Beaton, J., and Lo, K. (1991a), "Expert Systems for On-line Fault Diagnosis in Electrical Power Networks", Proc. Int. Conf. on Electricity Distribution, Liege, Belgium, 4.3.1 - 4.3.7

Cockburn, D. Corera, J. Cross, A. Echavarri, J. Laresgoiti, I., and Perez, J. (1991b), "Development of Two Large Industrial Applications Within A Distributed Artificial Intelligence Framework", ARCHON Technical Report 18, Atlas Elektronik, Bremen.

Cockburn, D. (1992), "Two Model Based Systems for Fault Diagnosis in Electricity Distribution Networks", IEE Colloquium on Intelligent Fault Diagnosis, Part 2: Model Based Techniques, Digest No. 1992/48, London, UK.

Cockburn, D., Varga, L. Z, and Jennings, N. R., (1992), "Cooperating Intelligent Systems for Electricity Distribution", Proc. of Expert Systems 92 (Applications Track), Cambridge, UK.

Cockburn, D. (1993), "Intelligent Network Operation and Control Systems" IEE colloquium on Expert Systems in the Field of Protection and Control, Digest No. 1993/193, London, UK.

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.

Cross, A. Brailsford, J., and Brint, A. (1992) "A KBS for writing Safe Sequences of Operations on a High Voltage Electricity Network", Proc. First Int. Conf. on the Practical Applications of Prolog, London, UK.

Jennings, N. R., (1992) "Using GRATE to Build Cooperating Agents for Industrial Control", Proc. IFAC/IFIP/IMACS Int. Sym. on Artificial Intelligence in Real Time Control, Delft, The Netherlands, 691-696.

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.

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.

Lees, M. (1992), "Measurement of Lightning Ground Strikes in the UK", Proc. Int. Conf. on Lightning Protection, London, UK, 2.1.1 - 2.1.5.

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

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

Scott, L. (1988), "A Lightning Location System for the UK Electricity Supply Industry", Proc. Int. Conf. on Lightning and Static Electricity, Oklahoma, 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.