Pergamon

Expert Systems With Applications, Vol. 7, No. 4, pp. 563–579, 1994
Copyright © 1994 Elsevier Science Ltd
Printed in the USA. All rights reserved
0957-4174/94 $6.00 + .00

Integrating Intelligent Systems into a Cooperating Community for Electricity Distribution Management

LÁSZLÓ Z. VARGA*

Department of Electronic Engineering, Queen Mary and Westfield College, University of London, Mile End Road, London E1 4NS, UK

NICK R. JENNINGS

Department of Electronic Engineering, Queen Mary and Westfield College, University of London, Mile End Road, London E1 4NS, UK

DAVID COCKBURN

EA Technology, Capenhurst, Chester CH1 6ES, UK

Abstract—*Systems in which semi-autonomous problem-solving agents communicate and cooperate with one another represent an exciting vision of future computing environments. However, if this vision is ever going to result in commercially viable systems, then consideration must be given to the large software base that exists within many organisations. Success requires the ability to incorporate pre-existing systems alongside purpose-built agents in a cooperating community. This requirement is vital because the former represent a substantial resource investment that companies cannot afford to consign to the scrap heap. We report on our experiences of constructing cooperating communities that contain elements that were pre-existing and some that were developed specifically for incorporation into an integrated environment. The general purpose framework of ARCHON (ARchitecture for Co-operative Heterogeneous ON-line systems) provides the underlying technology to facilitate cooperative problem solving, and the exemplar domain is the real world problem of electricity distribution management. The actual application being developed is called CIDIM (Cooperating Intelligent systems for DIstribution system Management). An evolving methodology for designing and developing a mixed system such as this is outlined, based on our experiences in CIDIM and several other real-world industrial applications. It specifies a hybrid top-down and bottom-up approach to integration, identifies the important characteristics that shape multi-agent problem analysis, and outlines key factors that impinge upon the design of the community. This methodology is then used to motivate the design decisions for the CIDIM application. Finally the process of instantiating the individual agents is discussed, some helpful guidelines on testing and evaluating future applications are given, and the implementation of one of CIDIM's cooperative scenarios is described in depth.*

1. INTRODUCTION

IN SOPHISTICATED industrial applications, there is a growing desire to intelligently integrate information from a diverse range of sources. Such information usually is produced from multifarious data-gathering and problem-solving activities, each of which is concerned with a particular (partial) aspect of the overall process. However, because of the interdependencies that exist through the common process, if information is shared in an intelligent and efficient manner the performance of both the individuals and the system as a whole can be enhanced. Consequently an increasingly large number of applications are being conceptualised in terms of cooperating agents; examples include speech processing (Erman & Lesser, 1975), flexible manufacturing systems (Parunak, 1987), air traffic control (Cammar-

* László Varga is on leave from KFKI Research Institute for Measurement and Computing Techniques, Budapest POB 49, 1525 Hungary, OTKA F4064.

Requests for reprints should be sent to Dr. N. R. Jennings, Department of Electronic Engineering, Queen Mary and Westfield College, University of London, Mile End Road, London E1 4NS, UK.

ata, McArthur, & Steeb, 1983), electricity transportation management (Jennings, Mamdani, Laresgoiti, Perez, & Corera, 1992), telecommunications network management (Weihmayer & Brandau, 1990), design (Klein, 1991), concurrent engineering (Reddy, Srinivas, Jagannathan, & Karinthi, 1993), sensor interpretation (Lesser and Erman, 1980), and particle accelerator control (Jennings, Varga, Aarnts, Fuchs, & Skarek, 1993).

In such Distributed Artificial Intelligence (DAI) systems, each agent is capable of some useful problem-solving activity in its own right, but by communicating and cooperating with others it is able to enhance its performance (Bond & Gasser, 1988; Gasser & Huhns, 1989; Huhns, 1988). Such agents have two distinct types of knowledge and reasoning capabilities: one for solving domain-level problems (e.g., fault diagnosis, security analysis, and restoration planning) and another for participating in environments containing other entities (e.g., being able to request information and services from acquaintances and sending timely information to help community members).

Jennings and Wittig (1992) highlight two features of industrial applications that have a significant impact upon the design of a cooperating community. First, there is a vast amount of pre-existing software currently being used. Second, a given application is composed of a diverse range of generic tasks (e.g., diagnosis, planning, monitoring, and scheduling). The upshot of these two observations is that any ensuing multi-agent system in this domain is heterogeneous at many distinct levels—different problem solving methods are employed (e.g., heuristic and model based diagnosis), different types of computer systems need to interact (e.g., expert systems, databases, and conventional numerical software), and different base-level programming languages are used (e.g., LISP, Prolog, C, FORTRAN, and Informix).[1] Having to deal with pre-existing and heterogeneous software meant that a range of new DAI problems needed to be tackled. These issues had been obscured in previous systems because they had dealt predominantly with homogeneous and purpose-built problem solvers. As such, the problems that were raised and the solutions that were developed during this work represent a necessary and important step in bridging the gap between the simplifying assumptions of academic exercises and real-world multi-agent systems.

This work contributes to the general body of information on DAI by describing a new application and how it was solved using multi-agent techniques. From an application perspective, we describe a novel and promising approach to tackling the complexity of the real-world problem of electricity distribution management—hereafter referred to as CIDIM (Cooperating Intelligent systems for DIstribution system Management) (Cockburn, Varga, & Jennings, 1992; Corera, Laresgoiti, Cockburn, & Cross, 1993). An evolving methodology for analysing, designing, and building multi-agent systems for industrial applications that contain several pre-existing software components is described (Section 3). After introducing the electricity distribution domain, these design guidelines are used to motivate the structure of the CIDIM community. The benefits of the resulting cooperating ensemble are then explained in terms of the improved system functionality, the greater automation of mundane tasks, and the reduced burden on the control engineer (Section 4). Finally the process of instantiating the individual agents is discussed, some helpful guidelines on testing and evaluation are given, and the implementation of one of CIDIM's cooperative scenarios is described in depth (Section 5). The agent community was constructed using the ARCHON (ARchitecture for Cooperative Heterogeneous ON-line systems) platform (Jennings & Wittig, 1992; Wittig, 1992), which is a general-purpose multi-agent framework for industrial applications (Section 2).

2. THE ARCHON ARCHITECTURE

ARCHON agents have two distinct components; an *Intelligent System* (IS) and an *ARCHON Layer* (see Figure 1). The former may be pre-existing or may be purpose-built and solves problems such as detecting disturbances, recording status information, and diagnosing faults. From the ARCHON Layer perspective, the IS is composed of a number of atomic tasks, although in terms of their actual implementation the tasks may themselves be relatively sophisticated problem-solving activities involving branching and decision making. The latter is a meta-level controller that operates on the IS to ensure that its activities are coordinated with those of the others within the community.

Communication between agents is via message passing and is controlled through the High Level Communication Module (HLCM). It is deemed High Level because it provides not only standard communication facilities (achieved through a Session Layer implementation) but also services such as intelligent addressing and filtering. For example, if the IS produces a result that is relevant for other agents, the Planning and Coordination Module (PCM) can simply instruct the HLCM to send it to all interested agents without having to enumerate them. A message-passing paradigm was chosen in preference to a shared memory approach [e.g., blackboard (Hayes-Roth, 1985; Nii, 1986a/b)] because it has well understood semantics and offers a more abstract means of communication (Hewitt &

[1] A more complete classification of the types of heterogeneity that are found in such applications and how they impact on the design of subsequent multi-agent systems is contained in (Roda, Jennings, & Mamdani, 1991).

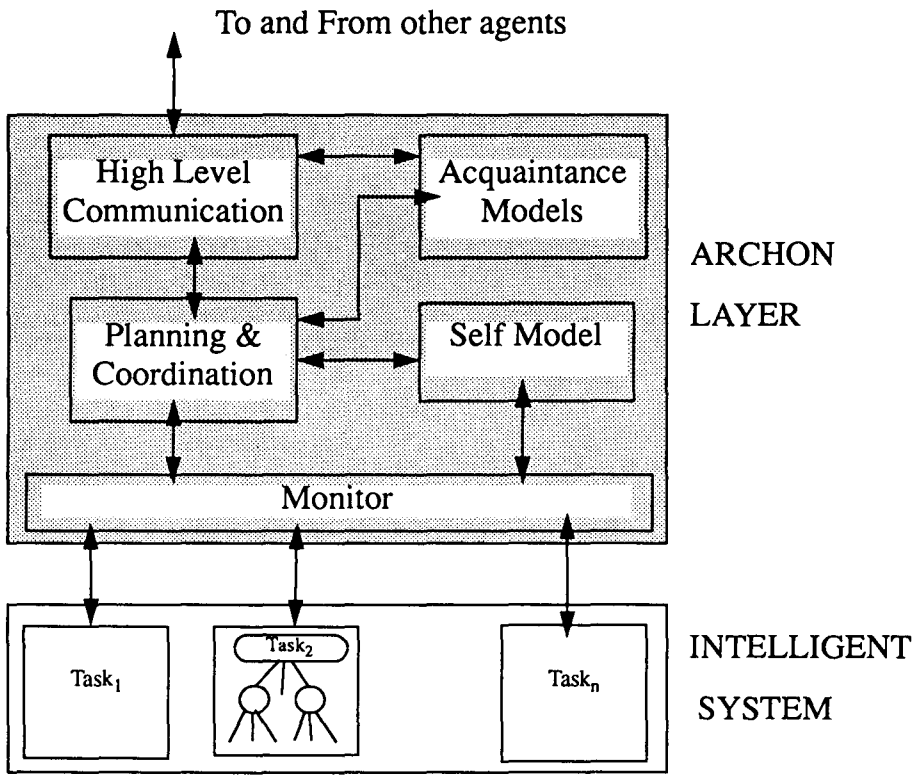

FIGURE 1. ARCHON Agent Architecture.

Kornfield, 1980); no hidden interactions can occur, so there is greater comprehensibility, reliability and control over access rights; it makes fewer assumptions about the system architecture. Also the physical distribution of the problem-solving agents and the desire to conform to the OSI standard for communication made message passing the natural choice.

The Agent Acquaintance Models (AAMs) are a representation of other agents in the community. Information maintained includes an acquaintance's skills, interests, current status, workload, and so on. These models are a prerequisite for coordinated activity because they provide a characterisation of the social problem-solving context where the agent has to operate (Jennings et al., 1992). The HLCM's intelligent addressing facilities also make use of these models to find the agents interested in a specific result. Much as the AAMs represent other agents in the community, the Self Model (SM) is an abstract characterisation of the agent's underlying IS. It contains information about the current state of the IS and embodies a representation of the sequences of actions that can be executed by the ARCHON Layer in its underlying IS.

The Monitor coordinates locally executable activities and is responsible for passing information to and from the IS. *Skills* are the coarsest granularity at which these activities are described. Other ARCHON Layer components (e.g., PCM, SM, and AAMs) deal exclusively on the level of skills, but within the Monitor they are given a finer structure—corresponding to a predefined OR-graph in which the named branches specify alternative solutions. Branches are composed of named units of activity called *plans* and are traversed in a depth-first manner. The associated constraints determine path selection at runtime. The nodes of the graph are called *monitoring units*, and they correspond to the invocation of individual tasks within the IS.

The PCM is the reflective part of the ARCHON Layer, reasoning about the agent's role in terms of the wider cooperating community. 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 that 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).

ARCHON's layered and modular architecture is ideally suited for industrial applications. It has been successfully applied to several problems, including electricity distribution (see Sections 4 and 5), electricity

transportation (Wittig, 1992, Chapter 8); cement factory control (Stassinopoulos & Lembesis, 1993); flexible assembly robotic cells (Oliveira, Camacho, & Ramos, 1991); and particle accelerator control (Jennings et al., 1993). The separation of the domain and cooperation know-how into the IS and the ARCHON Layer, respectively, allows pre-existing systems to be incorporated into the multiple agent community with relatively few modifications. Without this clear demarcation, extensive changes to the existing systems would be required to provide them with the necessary knowledge to interact with and benefit from the other agents in the community. Also, by providing a configurable mechanism for agent control and by defining a language for interacting with ISs, the Monitor masks the underlying heterogeneity of the domain system and allows the ARCHON Layer to operate upon a homogeneous representation.

3. METHODOLOGY OF INTEGRATION

3.1. A Two-Sided Approach to Integration

The instantiation of a multi-agent system can proceed in two ways. Using the concepts of a general purpose framework, an application can be built completely from scratch in a top-down manner. Alternatively, the elements that satisfy the application requirements are developed in a bottom-up manner. However, neither of these approaches is entirely satisfactory for the majority of industrial applications; the former means that already existing software is not used, the latter that no reusable components are available for subsequent use in future multi-agent applications. Thus, it was decided that successful integration for this class of problem requires a mixture of the two approaches to be applied simultaneously and iteratively (see Figure 2). Therefore, whilst constructing the agents, the application designers continuously compare their design with the general architecture and, if the concepts are being poorly utilised, a redesign may be necessary. The bottom-up forces ensure that this design is continually compared with the application requirements and constraints.

In our case, the top-down part of the integration process expresses the fact that the application designer must understand the ARCHON architecture and its associated concepts and then apply them to meet the problem requirements and the abilities of the already implemented elements. For CIDIM, the bottom-up aspect constitutes the bigger part of application design. It involves analysing the problem from a multi-agent perspective and extending or modifying the existing systems into a cooperating community so that the general framework can be applied in the most beneficial manner. An initial methodology for these bottom-up processes is discussed in the following subsections, and its application to CIDIM is described in Section 4. At this stage the methodology cannot be described in a fully formal manner nor can it claim complete generality because it has only been applied to industrial control applications. The current situation is somewhat akin to that of knowledge engineering for stand-alone expert systems; as experts cannot fully and formally describe the knowledge needed to solve their problems, they illustrate it through examples. The multi-agent analogue is that developers and end-users prefer to see examples of how cooperation can solve a problem in the application domain rather than abstract examples of different cooperation types. Despite this informality, a methodological guide with such application-derived examples is still extremely helpful in structuring their approach to system design.

3.2. Multi-Agent Problem Analysis

Designing a cooperating community requires that the problem be analysed from a multi-agent perspective. The following key issues are dealt with in the remainder

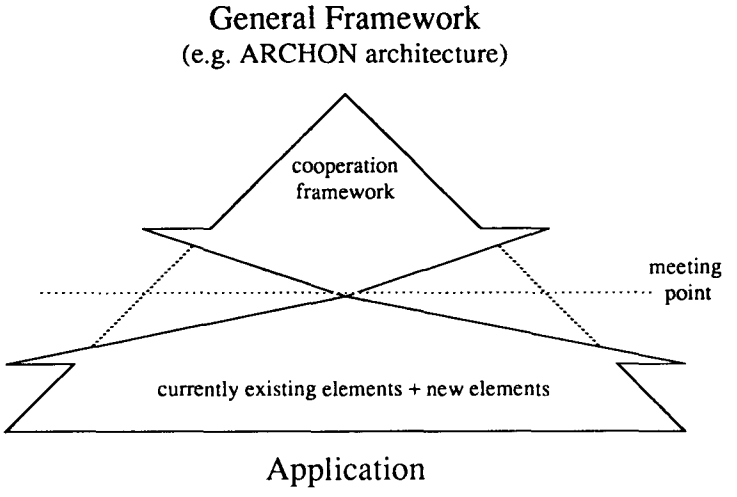

FIGURE 2. Integration methodology: Meeting in the middle.

of this subsection: analysis of pre-existing systems; analysis of the cooperation amongst the operators controlling the process; analysis of possible means of decomposing the systems; plans for new system components, and the detection of new cooperative situations made possible by system integration (see Figure 3).

3.2.1. *The Pre-Existing Intelligent Systems* The capabilities of the existing ISs have to be identified. Questions that need answering include "What is their main function?," "How is this function achieved?," "What tasks are utilized in the operation of the IS?," "Are these tasks executable in a stand-alone way, or are they deeply embedded in the IS?," and "Are there tasks or data that are duplicated in several ISs?." This analysis must also address the issue of the user interface. Most existing ISs have an interface, which allows the operator to interact with their component tasks (see interactions a). However, some portion of these interactions are replaced by cooperation amongst the agent community (see interaction b). For those interactions that cannot be replaced by interagent communication, some of them can be inherently (e.g., derived from the structure of the company) attached to the IS, and some of them can be moved to a separate interface agent. These three types of interactions must be identified and classified and appropriate modifications and/or extensions made to the interfaces.

3.2.2. *Cooperation between the Operators.* In many industrial applications, the partial implementation of various facets of system control requires the operators to interact with one another to produce effective overall responses (Jennings & Wittig, 1992). Analysis of these interactions (shown as c) gives an important insight into some of the types of cooperation that are likely to occur in the multi-agent system implementation. Factors that need to be identified include the data and activity requests communicated between the operators; the criteria that determine when requests are issued and to whom; the mechanism by which requests are issued; the protocol adopted for the request and how the result is accepted. Sometimes communication is not apparent, because two systems are operated by one individual; in this case the information transfer must be brought into the open.

3.2.3. *Possible Decompositions of the Intelligent Systems.* Some large ISs are difficult to maintain in their own right, and introducing a further level of complexity by incorporating them directly into a cooperating community would simply be infeasible. In such cases, it is worth splitting up the IS into several agents. Another reason for decomposing an IS into multiple agents is if it has a capability that is replicated by another component that will become part of the community. In this situation, a decision has to be reached about the tradeoff between the efficiency of creating a dedicated agent and the desirability for robustness, which comes from duplicated capabilities. There is an analogous situation for data: if the same information is needed by several agents, then it is often worth introducing a database agent to maintain it. This is especially true if the data is a description of the process controlled by the ISs, because it can be kept up-to-date and consistent more easily.

In terms of Figure 3, a pre-existing system has been split up—most of it remains in system A, but an identifiable portion (B) has been combined with another piece of pre-existing software (C) to make the intelligent system of the middle agent.

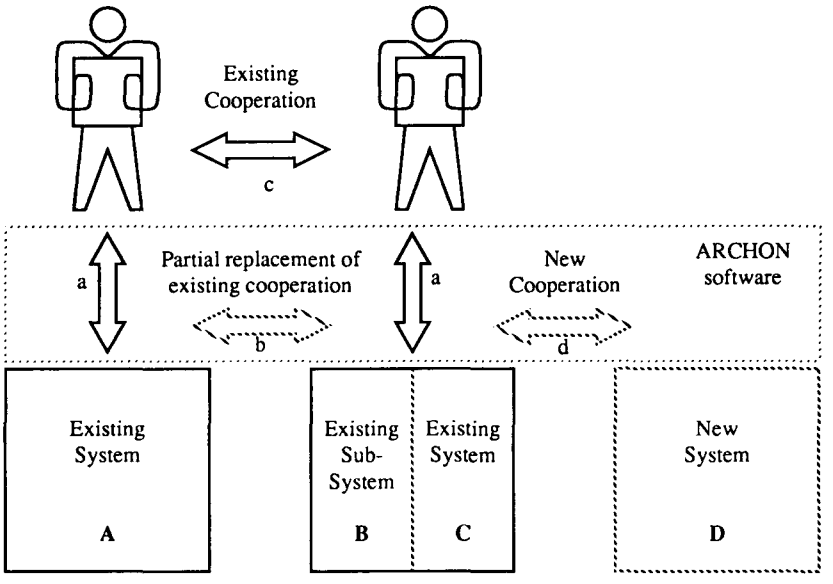

FIGURE 3. Problem analysis from a multi-agent perspective.

3.2.4. *Introducing New System Components.* Some components of the application may not be automated at the time when the multi-agent system is being evaluated. In this case it is worth investigating the benefits of introducing new ISs to deal with these activities (see system D in Figure 3). New ISs may also be added for reasons other than missing functionality. For example, if a particular IS is often overloaded, then this bottleneck may be avoided by making it into several identical agents within the community. To enhance solution quality, new ISs can be added that generate the same information as an existing system but that work with different data and different side effects.

Newly written ISs can take advantage of being designed specifically for use in an agent community by having the ability to use information from acquaintances that would not be available to them outside of the particular social context. For example, in CIDIM a purpose-built diagnosis agent makes use of lightning location information in its reasoning process because an acquaintance is able to provide it. Outside of CIDIM this information would not normally be available. To maximise software reusability and system robustness, the diagnosis agent is not dependent on the lightning location information, but if it is available then it can be used to improve the quality of its output. The same use of optional additional inputs by existing ISs can sometimes be achieved. However, this always involves some restructuring work, because the additional data would not have been accounted for in the original design.

3.2.5. *Introducing New Forms of Cooperation.* Sometimes in nonintegrated systems, potential interactions that would be beneficial to the community's problem-solving efforts are not undertaken because they would require a substantial amount of manual effort or cross-checking. In these cases (interaction d in Figure 3), an integrated system offers significant improvements by automating these exchanges.

3.3. Agent Community Design

Once the problem analysis has been completed, design of the multi-agent system can commence. To be successful this design must consolidate the valuable features of the existing system, improve on some of the existing features, and introduce new features by exploiting the opportunities provided through social interactions. By adopting a mixed approach to integration, community design imposes more constraints than designing a system from scratch, because the designer should aim to maximise software reuse. However, this should not be taken to the extreme, in certain cases it is better to cease using certain parts of the existing system and change to new ideas brought in through the multi-agent system approach.

The first phase of system design concerns the community. Issues that need to be addressed include determination of the agent granularity, identification of the role of each agent, design of the user interfaces, definition of the agents' skills, and enumeration of the forms of inter-agent communication. Once the community design has been completed, each individual needs to be instantiated (see Section 5 for details of this process).

3.3.1. *Level of Granularity.* The first step is to determine the granularity of the agents. In many cases, it is the same as that of the existing systems because it involves the least amount of restructuring work. However, the granularity may be changed if the application is growing and needs redesign or if the granularity does not really fit the problem. For example, earlier experiments in the domain of particle accelerator control showed that mapping two pre-existing expert systems directly into a community of two agents did not allow the full potential of the cooperating systems metaphor to be exploited (Jennings et al., 1993). Only by decomposing the two agents into a community of six could the benefits of parallelism and asynchronous interactions be fully realised.

Other factors that impinge on decisions about granularity are the limitations on agent size caused by the availability of computing resources; the natural distribution and structure of the problem, and the ease of transforming (restructuring) the existing systems.

3.3.2. *Role of Each Agent.* The next step is to determine the functional role of each agent and of the community as a whole. If as a result of the problem analysis phase it is determined that the existing ISs fulfil all the necessary requirements, then the role of the agent community is basically the same as that of the original system. In this case, the ISs only need to be extended with the appropriate cooperative functions. However, if the existing ISs do not fulfil all the requirements, then identifying the role of each agent is a more time-consuming task. The functionality of the whole application has to be described, this functionality has to be broken into identifiable subproblems, these subproblems are then mapped to IS tasks and the IS tasks have to be allocated to ISs. All of this has to be undertaken while still being mindful of the constraints imposed by the structure of the existing ISs.

3.3.3. *User Interface.* In the same way as each agent's role is specified, so the role of the user interface(s) also has to be determined. The basic question is whether the designer wants to hide all the agents behind one interface and make the cooperating community transparent or whether each agent has its own interface and the community and its interactions are made visible (Avouris, 1992). A single interface is most appropriate

when the application is centred around one problem or when one operator can survey the whole application. Multiple interfaces better fit those applications in which agents mainly serve their own goals and cooperation manifests itself as occasional assistance.

3.3.4. *Agent Skills.* Once the functional role of each agent has been defined, the skills to be represented at the ARCHON Layer have to be decided upon. For newly developed ISs, the designer has a large degree of freedom in deciding upon the skills. With pre-existing systems, the process may be severely constrained by the current structure of the system. Skills are allocated so that agents are able to fulfil their designated role and also so that they represent the largest possible granularity of action. The latter point ensures that they are large enough to warrant the overhead inherent in skill execution, yet sufficiently small to concisely fulfil the desired objective. The mandatory and optional inputs of the skills must be described as well as the intermediate and final results that are produced.

It should not be possible to combine two skills without losing something in the role of that agent, and conversely, it should not be necessary to divide a skill into subskills unless this enhances the role of the agent. Thus, for example, an application designer may be faced with the decision of whether to make tasks A and B of an IS available as skills in the ARCHON Layer or whether to combine them into a single skill. This may be the case when in the normal operation of the IS function B always follows function A. In making this decision, the following points should be considered: if the answer to any of the following is "yes," then A and B should be separate skills. If the answer is "no" and it is likely to remain this way even with future developments in the application, then for efficiency reasons it is better to combine them into a single skill.

- Will the calling of B ever be contingent on the output of A, and, if so, will the ARCHON Layer be able to assist in evaluating the decision?
- Could the calling of B be dependent on conditions in another agent?
- Is A or B an activity that another agent might want to request as a service (e.g., an acquaintance wants B performed, but not A)?
- Are there different requirements regarding concurrent execution for A and B? If many versions of A can run concurrently but B is a critical section it makes sense to have separate skills and use the locking mechanisms of ARCHON for B.

3.3.5. *Messages.* From knowing the skills that are present in the community, it is possible to describe what types of messages are sent between the agents, what format the information is in, and what the expected size of the transmissions are. Examples of generic message types supported by ARCHON include requests for data, requests for the execution of skills, spontaneously volunteered information, responses to requests, reports of current status, execution failure messages, and emergency status messages.

It is also important to decide the mechanisms that should be employed to disseminate useful information around the community. For example, if an agent is capable of producing a result that two of its acquaintances are interested in only if they are in specific problem-solving state, then this data cannot be volunteered as unrequested information (because the sender cannot be constantly aware of the exact state of the other community members) and should be requested by the agents when they are in the relevant state. In other cases, the main form of social interaction is through the spontaneous sending of information that is believed to be relevant to the recipient.

4. COOPERATIVE ELECTRICITY DISTRIBUTION MANAGEMENT

4.1. Introducing Electricity Distribution Management

Electricity distribution systems deliver electrical energy from the transmission substations to customers, transforming to a suitable voltage where necessary. In the U.K., Regional Electricity Companies (RECs) are responsible for distribution and supply to customers within a geographical area. Most electricity is supplied by the generating companies via the national grid at 400kV and 275kV to grid points where it is fed through transformers to the 132kV network of the REC. From here it is passed around at a variety of voltage levels, to different types of customers (see Figure 4).

Management of the distribution network aims to ensure that there is a secure supply to all customers. This involves maintenance and repair of plant, reconfiguration of the network for stable operation, and restoring supply that has been lost due to faults. Coordination within and between the different voltage levels is carried out at a control centre by control engineers (CEs) who aim to maintain the optimal network con-

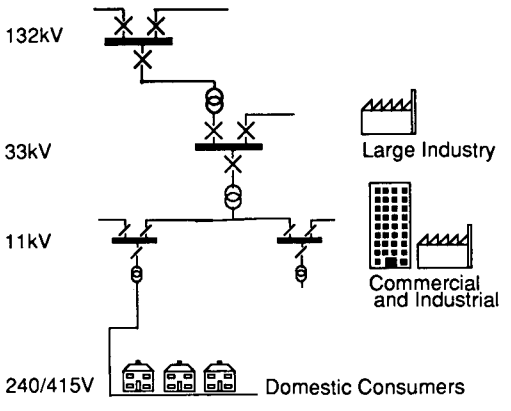

FIGURE 4. The distribution network.

figuration, which keeps all customers supplied and minimises losses. Listed next are several forces that can cause the network to deviate from its steady state and that the CE must therefore take into account when making decisions about which actions to perform. Some of these forces can be anticipated well in advance, meaning that a schedule of actions can be agreed on beforehand; others are unplanned and require the engineers to create appropriate strategies in real time.

- *Weather:* Temperature fluctuations cause changes in load. Wind, icing, and lightning damage the network.
- *Demand:* Demand changes according to the time of day, the day of the week, and the weather.
- *Events:* Third party damage, equipment failure, industrial action, disasters.
- *Planned work:* Maintenance, installing new equipment, reinforcement.

The higher voltage networks are usually managed solely from the control centre, meaning that authorisation must be given for any actions to be carried out by a field engineer and that the outcome must be reported back. With the lower voltage levels, the field engineer may be authorised to perform a sequence of operations to a preplanned schedule and only report back on significant items. Figure 5 represents a schematic picture of a control centre's typical communication flows.

The extent to which the CE directly controls the network, based on the measurements, indications, and alarms available to him/her and through the use of telecontrol command schemes, varies between RECs. However, in all cases it can be an extremely difficult task; decisions have to be taken based on incomplete information, presented from a variety of diverse sources, and the resulting actions must be both timely and secure. The major activity is to grant authority for switching operations to the field engineers (except for the 240/415V local network) and, where telecontrol equipment is installed, to initiate the remote operation of plant via a console in the control room. These activities are the outcome of the CE's reasoned decision-making process and are a response to his perception of the network's needs:

- If the network is quiescent, work on it consists of repairs and planned routine maintenance, which requires authorisation for, or actual performance of, the necessary switching operations to release the plant from the network.
- If there is a fault in the network, responsibility has to be assumed for initiating remedial action.
- The network configuration needs to be continuously monitored to ensure that it is in a stable electrical condition. This includes assessment of the load on the network to respond to overloads on plant items or to take early corrective action to avoid it: actions to avoid excessive fault levels and actions that aim to improve system security within the constraints of the current network status.

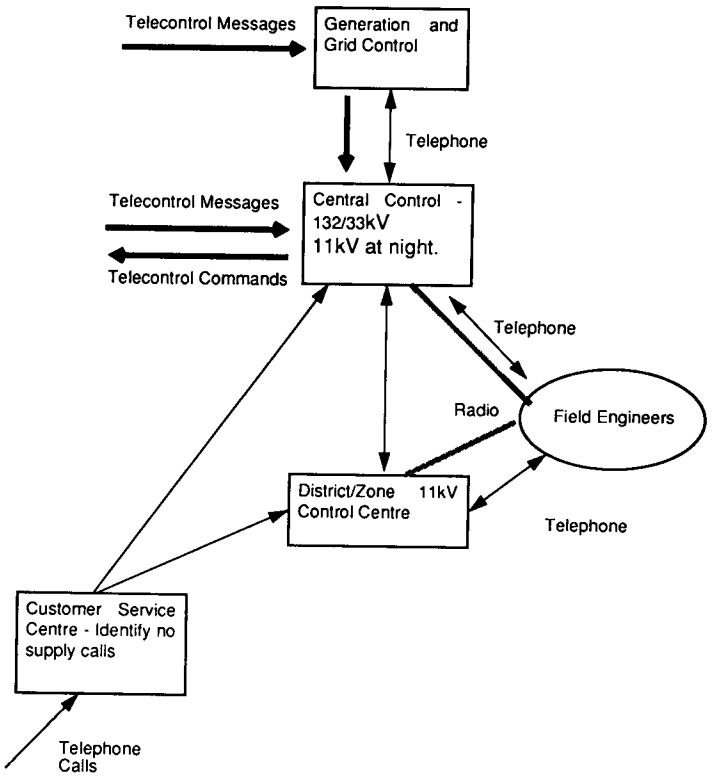

FIGURE 5. Communication flows in the control centre.

4.2. Analysis and Design of the CIDIM Community

Following the methodology outlined in the previous section, the first task when developing the CIDIM application was to analyse the problem of electricity distribution management. From the top-down perspective, a specification of the whole CIDIM application was developed. This necessarily included the functionality already available in the ISs, but also highlighted some of the functionality that could not be assigned to any single IS (i.e., those functionalities that require cooperation between ISs or those that utilise ARCHON Layer concepts). From the bottom-up perspective, analysis involved a detailed examination of the pre-existing ISs, identification of the types of cooperation that take place between CEs, study of the possible decompositions of the ISs, identification of potential new components and recognition of any new forms of co-operation that could be realised through integrated problem solving.

In this application several pre-existing systems were available to assist the CE in managing the network. These included a switching schedule production assistant (SSPA) (Brailsford, Cross, & Raven, 1987; Cross, Brailsford, & Brint, 1992 & 1993); a weather watch expert system that locates lightning strikes (Lees, 1992; Scott, 1988) and a high voltage diagnosis expert system (HVDES) (Bramer, 1988; Cockburn et al., 1991, 1992). These systems formed the basis of the CIDIM application, but needed to be transformed into a cooperating community using the established methodology.

The HVDES had a separate program to translate and filter the telemetry messages arriving from the electricity network. Because other agents also need this telemetry, a dedicated Telemetry Agent (TA) was created to handle telemetry for the whole system (see Section 3.2.3). Similarly, the network model of the HVDES was self-contained. However, as each REC represents its network model in its own database, using its own format, it was deemed appropriate, on the grounds of consistency (see Section 3.2.3), to create an information agent (IA) to provide a front end to the actual database. This organisation also allows easier access to the network models, and should the database need to be changed, for example, using CIDIM in a different REC, the alterations are confined to the internal workings of the IA. From an external perspective, the format of the queries posed by other agents and the response from the IA will remain the same.

The HVDES does not cover the whole distribution network, so to produce a more comprehensive aid to the CE, a low voltage diagnosis expert system (LVES) (Cockburn et al., 1992) was developed (see Section 3.2.4). The LVES was conceived as separate from the HVDES because its input data is different and because its diagnosis method is necessarily different (incremental refinement rather than simulate and test). The reasons for these differences are, first, that the operation of automatic protective devices only locates a fault to the circuit level in the low voltage network and, second, that important additional information is obtained from customer telephone calls that report loss of supply.

For this application it was deemed appropriate to develop a standard presentation system for information to the CE (see Section 3.3.3). This Advisor Agent (AVA) allows information derived from more than one source to be presented in a standard way. For example, the CE is interested in fault reports, not in the fact that two separate agents were responsible for producing them. Also, with an integrated presentation, lightning information can be displayed next to faults that it may have caused, thus providing a useful cross-check of information for the CE (see Section 3.2.5).

In addition to diagnosing faults, CEs spend a significant proportion of their time on routine maintenance and restoration of power. However, before maintenance can begin, a plan of safe switching operations needs to be made that isolates that part of the network to be worked on. The pre-existing SSPA was able to perform this job, and by integrating it into the CIDIM community, it could use the network data from the IA (previously it had its own source). The SSPA has its own graphical user interface with which the CE interacts to produce switching plans. In CIDIM these plans can be made available to other agents so that checks can be carried out to determine whether newly detected faults will interfere with them. If this is the case, such interference will be signalled to the user via the AVA. Thus the SSPA can use its own specialised interface for creating switching plans, so to the user it appears the same as it does in stand-alone mode (see Section 3.2.1), and the AVA can be used for giving warnings about new faults affecting switching plans. The latter is additional functionality brought into being by the SSPA's being included in CIDIM.

A switch checking agent (SCA) will also be included in CIDIM. This agent will have the same safety checking capabilities as the SSPA but will not require the separate user interface. Its role will be to recheck already created plans against the current network status before they are carried out. This recheck will be automatically triggered by the ARCHON Layer, and the results will go to the AVA (see 3.2.5).

When a fault occurs it may be that no customers actually lose supply—because of alternative routes through the network. These new routes can be switched in by the CE after he has checked, using the SCA, that they will not damage the network. This new network configuration might work temporarily but may leave the distribution system in a precarious state with a line overloaded or in a state such that a subsequent fault may cause the loss of power to a far greater area. Thus a Security Agent (SA) was included to check for overloads and to consider the network security should sub-

sequent faults occur. Previously such programs were not used on line, but integration within CIDIM means that an up-to-date network model is available (through the IA) and that the checks can be triggered to run automatically at suitable times (e.g., after a fault; see Section 3.2.5).

In comparison to the majority of DAI applications, the level of granularity of the CIDIM agents is high. The reason for this is that in CIDIM the ISs stem from stand-alone systems that were themselves of high complexity, whereas most DAI systems simply build their applications from scratch without having any bottom-up design constraints. Obviously, CIDIM's distinct functional components could be made into separate agents, as with the HVDA, but finer decomposition would require significant re-working of the remaining structure and would therefore contravene the basic principle of software reusability. A further constraint is that several of the pre-existing systems will continue to be used in stand-alone mode outside CIDIM. Therefore, if the two versions of the software are substantially different, then there is a significant extra burden on maintenance.

4.3. Cooperation in CIDIM

One of the major benefits of interaction in this application is to automatically collate results between different agents and to ensure that timely and consistent information can be accessed by all community members. For example a switching plan for a routine maintenance operation may be made one day to be carried out the next. However, if a fault occurs in that area in the meantime, the plan will have to be redone. Within CIDIM all the necessary information is dispersed at various sites within the community, but by instantiating the necessary social interactions these checks can be carried out automatically. Two other types of cooperative interaction that enable the multi-agent community to perform more robustly in the face of missing information and agent failures are detailed next.

4.3.1. *Assistance when Information is Missing or Unavailable.* The distribution network is protected from faults by automatic circuit breakers. The TA, the high-voltage diagnosis agent (HVDA) and the low-voltage diagnosis agent (LVDA) can all make assumptions about the state of these circuit breakers. As the TA receives telemetry messages when circuit breakers operate, it is usually assumed to have the definitive view of the network's state. However, there are three exceptional situations where this is not the case: not all circuit breakers are telemetered; telemetry can go missing and not be received at the control centre; telemetry from a whole substation may not be reported for a period of time.

In each of these exceptional cases, the HVDA and LVDA can make assumptions about the network state, through their own knowledge and by interacting with their acquaintances and can convey them to the IA so that a more accurate description of the network is available to the community. First, the HVDA may use telemetry from the TA and its knowledge of how the protection system operates to hypothesise faults as the cause of the telemetry. It can then simulate them on its network model and show that a circuit breaker should have operated but that no telemetry message was received. Second, the LVDA can combine telemetry from the TA with telephone calls from customers reporting loss of supply in its diagnosis. These two sources of information can be used by the LVDA to indicate that a circuit breaker is open even if the corresponding telemetry has not arrived. Finally, an example involving a more elaborate pattern of interaction occurs when there is a high voltage fault that would mean a loss of supply to the low voltage network if it were permanent. However, the HVDA is uncertain about the state of a circuit breaker and, thus, the nature of the fault, due to missing telemetry. Here the HVDA can interact with the LVDA and the TA, asking the LVDA if power has been lost in the low-voltage network and asking the TA to reconfirm its opinion of the state of the circuit breaker (since a communication failure could have occurred in the telemetry system and the missing telemetry could have been received late).

4.3.2. *Cooperative Query Processing.* The LVDA can improve its diagnosis for overhead line networks by cooperating with the weather watch agent (WWA) to determine whether there was lightning at a certain time near a certain plant item. If there was lightning in the vicinity, then it is extremely likely that it was the cause of the fault. However, the WWA is unable to answer this question on its own because its database only contains information relating the time of lightning strikes to geographic locations. To determine whether the strike was near a particular item of plant, the WWA needs to interact with the IA, which is able to map between items of plant and geographic locations. The WWA can determine for itself whether there was lightning at a particular time. If the answer is "no," then there is no need to seek assistance from the IA and the LVDA's query can be answered. If the answer is "yes," then the WWA will interact with the IA to determine whether there was lightning near the particular item of plant in question and, when this answer is available, return it to the LVDA, which can incorporate it into its diagnosis.

This community design is beneficial in that the pre-existing IS of the WWA can stay as a general purpose system for lightning location and does not need to be augmented with knowledge about electrical plant.

Through cooperation with another agent (the IA), the WWA can be used in a specific domain (electricity distribution networks) without decreasing the flexibility of the original program.

5. INSTANTIATING THE AGENTS AND BUILDING A COMMUNITY

With the community's overall structure designed and each agent's role specified, the next task is to instantiate the ARCHON Layer of the individual members. The agent's ARCHON Layer then needs to be connected to the IS to produce a complete agent. Once all of the agents are completed, they need to be combined into a community and tested. The present implementation concentrates on the LVDA's diagnosis phase. Particular attention is given to the diagnosis, using telemetry received from the TA and the interaction with the IA to provide the appropriate model of the network.

5.1. Agent Instantiation

The functional role of each agent devised in the community design phase is made operational by defining each agent's skills. For example, the role of the LVDA is to diagnose faults in the low-voltage network. To carry out this task successfully, it needs information from the TA (DEAL_WITH_TELEMETRY), the IA, the WWA, and the HVDA (ACCEPT_HV_FAULT), as well as from direct telephone call inputs from customers (DEAL_WITH_TELEPHONE_CALLS). The telemetry coming from the TA may be delayed or lost (DEAL_WITH_SUBSTATION_FAILURE), meaning that the final diagnosis may take longer than expected. The result of the diagnosis is only made available to the CE after several telephone calls and telemetry messages have been received or after a fixed deadline has expired (REPORT_DIAGNOSIS). Using this as a basis, the following skills were assigned to the LVDA:

DEAL_WITH_TELEMETRY: Receive telemetry from the TA whose operation is autoreclosure (the circuit breaker opened and then reclosed automatically), create a network model for the area identified by the telemetry (using information from the IA), and refine the diagnosis.

DEAL_WITH_TELEPHONE_CALLS: Receive information about telephone calls from customers, create a network model for the area identified by the telephone call (using information from the IA), and refine the diagnosis.

REPORT_DIAGNOSIS: After sufficient time has elapsed so that no more information about a particular fault is expected, contact the WWA for information about lightning strikes in the area and produce a final report for the fault.

ACCEPT_HV_FAULT: Accept information from the HVDA about the occurrence of high voltage faults that may affect the low voltage network and use this information to refine the diagnosis.

DEAL_WITH_SUBSTATION_FAILURE: Receive reports of failure in the telemetry system and refine the diagnosis to take into account the missing information.

Once an agent's basic units of activity are defined, the local and social problem-solving behaviour required for a given application must be encoded through appropriate instantiation of the ARCHON Layer mechanisms. There are two main areas that need to be addressed. First, the agent's local problem-solving behaviour and its data definitions need to be encoded into the Monitor and the Self Model. Second, all the agent's social interactions and global situation assessment capabilities need to be represented through appropriate instantiation of the PCM, the Self Model, and the Acquaintance Models.

5.1.1. *Instantiation of Local Problem-Solving Activities.* To instantiate the necessary local control, the application designer has to encode the skills. Figure 6 shows a portion of the LVDA's DEAL_WITH_TELEMETRY skill, which accepts telemetry information and starts a fresh diagnosis activity if it corresponds to a new fault, or updates an existing diagnosis if it represents an area that is already under investigation. For simplicity, only one branch is elaborated. This has two named subcomponents: GET_NETWORK_DATA and DIAGNOSE_FAULT. The former ensures that the relevant portion of the network is loaded, and the latter actually performs the diagnosis based on the telemetry and the loaded network model.

The leftmost branch of the skill is traversed first. This means that the first step to be executed is the GET_NETWORK_DATA plan, which is itself composed of an OR-Graph of monitoring units (MUs):

```
PLAN GET_NETWORK_DATA
  :MU              MU_NEED_NETWORK
  :PRECONDITIONS   NIL
  :CHILDREN          (MU_GET_NETWORK  :END)
  :MU              MU_GET_NETWORK
  :PRECONDITIONS     ((:output-match NETWORK_DESCRIPTOR))
  :CHILDREN          (:END)
```

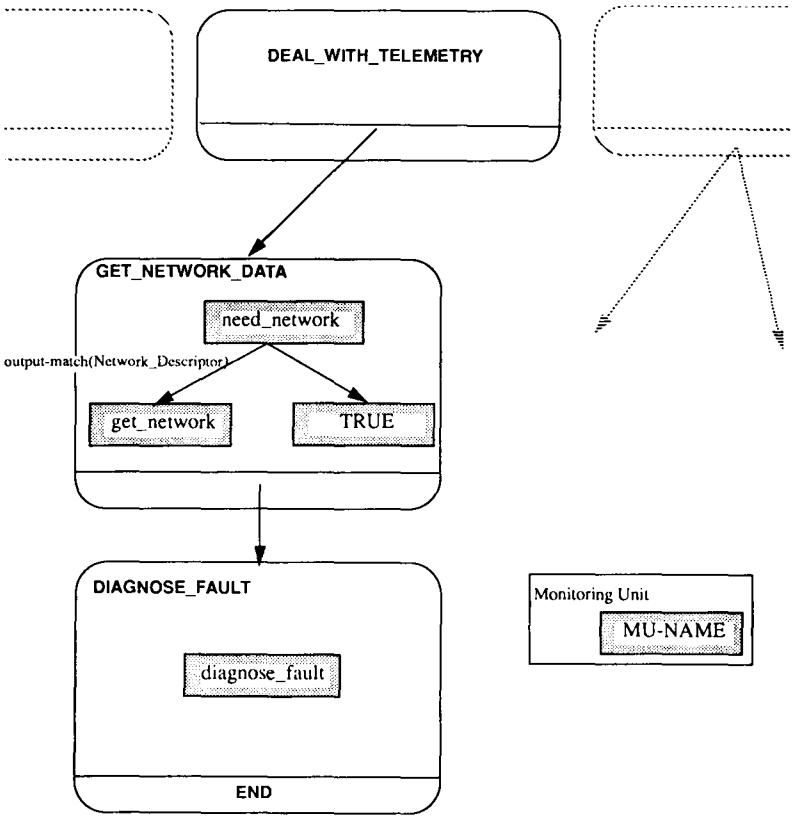

FIGURE 6. Specification of DEAL_WITH_TELEMETRY skill.

The root MU MU_NEED_NETWORK is executed immediately because its constraints on execution (i.e., preconditions) are satisfied (in fact, there are none). This MU invokes a function in the LVES called TASK_NEED_NETWORK, which is called with the parameters TELEMETRY (see below). Upon completion, this task returns a NETWORK_DESCRIPTOR, e.g., (:message-type NETWORK_DESCRIPTOR :content ((SUBSTATION SUB1) (VOLTAGE 11000)), which describes the network needed by the LVES or FALSE if the relevant portion of the network has already been loaded.

```
(define-MU-type
  :name MU_NEED_NETWORK
  :is-task TASK_NEED_NETWORK
  :parameters (TELEMETRY)
  :outputs (NETWORK_DESCRIPTOR))
```

When the plan's first action is completed, its children slot (MU_GET_NETWORK :END) is examined to identify the next activity to be performed. In this case the constraint on the execution of the leftmost branch (i.e., MU_GET_NETWORK) is examined. If these preconditions are met (i.e., the output of the previous MU is a NETWORK_DESCRIPTOR), then this step is followed. If its conditions are not met, then the next child is examined. Here it is the keyword :END that always succeeds.

Assuming that the NEED_NETWORK MU returns a network descriptor and then a new portion of the network needs to be loaded. Therefore, the GET_NETWORK MU will be executed:

```
(define-MU-type
  :name MU_GET_NETWORK
  :is-task TASK_CREATE_NETWORK_MODEL
  :parameters ( (NETWORK-DATA :para NETWORK_DESCRIPTOR) )
  :output NIL
```

Before calling the task in the IS, the Monitor ensures that all of the necessary input parameters are available. In this case, data of type NETWORK-DATA is not available in the environment of the skill, and so the Monitor signals to the PCM that this piece of information is needed and provides the NETWORK_DESCRIPTOR as the necessary parameters for computing the information. The PCM then determines how this information can be produced and by

whom (a process described in more detail in the following section). The PCM identifies that the best means of producing the desired information is to interact with the IA, and so it sends out the following request: (INFO-REQUEST NETWORK-DATA :parameters ((SUB-STATION SUB1) (VOLTAGE 11000)). The IA will eventually provide the necessary information to the LVDA's PCM, which will signal to the Monitor that it can restart its execution of the relevant MU. The CREATE_NETWORK_MODEL function of the IS is now invoked with the NETWORK-DATA returned from the IA, and the relevant portion of the network model is constructed in the LVES. Having finished this second plan step, its children are evaluated. However, as the CHILDREN slot contains :END there are none, and so the plan step has successfully completed.

As the GET_NETWORK_DATA plan has successfully completed, the next plan step is activated. DIAGNOSE_FAULT is a plan composed of just one MU:

```
PLAN DIAGNOSE_FAULT

  :MU              MU_DIAGNOSE_FAULT
  :PRECONDITIONS   NIL
  :CHILDREN        (:END)
```

The plan step DIAGNOSE_FAULT is executed because its preconditions are satisfied. This results in the LVES's DIAGNOSE_FAULT task being invoked with the TELEMETRY that has been provided from the TA as input (see below). This task first checks to see whether the supplied TELEMETRY relates to an existing hypothesis. If it does not, then a new hypothesis is created and a timer is started. The length of time spent on diagnosis depends on how soon the LVES expects to receive all the data that is normally produced when such a fault occurs (e.g., telephone calls from customers who have lost supply). This delay before the diagnosis is reported is necessary because all of the information coming from the customers will take a while to start arriving. If the TELEMETRY relates to an existing hypothesis, then the new information is used to refine the current hypothesis. In the meantime, other skills (e.g., DEAL_WITH_TELEPHONE_CALLS and ACCEPT_HV_FAULT) can be running and providing further refinements to the diagnosis. When the timer expires, the diagnosis is returned.

```
(define-MU-type
  :name   MU_DIAGNOSE_FAULT
  :is-task   DIAGNOSE_FAULT
  :parameters   (TELEMETRY)
  :outputs   (LVDA_DIAGNOSIS))
```

When this MU has completed, the result LVDA_DIAGNOSIS will be produced. There are no further plan steps and so the DIAGNOSE_FAULT plan has finished. As the last plan in the current branch of the skill has succeeded, the skill has been performed.

Once the skill has been defined, it needs to be placed into the Self Model so that other ARCHON Layer components can reason about it. For example, the DEAL_WITH_TELEMETRY skill is defined in the following manner (the triggering condition is dealt with in section 5.1.2):

```
Name:        DEAL_WITH_TELEMETRY
Trigger:     TELEMETRY-TRIGGER-FOR-LVDA
Inputs:      ((TELEMETRY  :mandatory))
Results:     (LVDA_DIAGNOSIS)
PlanName:    PLAN_GET_NETWORK_DATA
Children:    (DIAGNOSE_FAULT)
```

In many industrial applications, the data to be exchanged between agents has a complex structure. This structure must be specified in the Self Model and must be commonly understood by all the agents in the community. The structure identifies the constituent components of the data and defines their type. For example, the TELEMETRY data used above is an instance of the class TELEMETRY-MESSAGE-TYPE, which has the following definition:

```
type_create:  TELEMETRY-MESSAGE-TYPE
components: ( (TEXT:  STRING)  (TIME:  INTEGER)
              (TEXT_TIME:STRING)  (SUBSTATION:  STRING)
              (PLANT:  STRING)  (OPERATION:  STRING)
              (SOURCE:  STRING)  (VOLTAGE INTEGER) )
and a typical example is
  ( (TEXT  "ALARM  04SEP90  15.50/12.00  RUSKI  RUSKINGTON  C2
                  CIRCUIT BREAKER AUTOREC")
    (TIME 10874390)                    (TEXT_TIME "04SEP90 15.50/12.00")
    (SUBSTATION "RUSKINGTON")          (PLANT "RUSKINGTONC2")
    (OPERATION "AUTOREC")              (SOURCE "ALARM")
    (VOLTAGE 11000) )
```

5.1.2. *Instantiation of Social Problem-Solving Activities.* Analysis of CIDIM's social interactions reveals that the PCM is required to undertake the following duties: initiate cooperative interactions; select between local and remote execution of skills; if remote execution is selected, then choose the appropriate agent; if local execution is selected, then choose the appropriate skill (several local skills may provide the desired result) and service external requests. The reasoning required to perform these functions is built into the PCM's generic rules, and the application-specific details are contained in the Agent Models. Therefore, instantiating the necessary control of social activities mainly involves setting up the appropriate agent models. The only feature of the generic rules that can be modified is the relative priority of the various social functions: thus, an agent can be made service-oriented by increasing the priority of external communication or computation-oriented by increasing the priority of local activities.

With regard to the LVDA's diagnosis using telemetry, the PCM of the respective agent is responsible for the following functions:

1. *When the IS of the TA generates telemetry, the PCM must recognise, through its acquaintance models, that the LVDA is interested in receiving it. Knowing this, the PCM should volunteer the data as unrequested information.*

Interest descriptors indicate information that other agents would benefit from receiving. They allow the developer to describe under what conditions information should be sent to an acquaintance, thus reducing unnecessary traffic. This specification contains two parts, the name of the data item and the condition in which the acquaintance is interested in it. For example, the TA's model of the LVDA specifies that it is interested in receiving TELEMETRY data that refers to the low-voltage network (i.e., less than or equal to 11,000 volts). The "!!" and "?X" symbols in the condition are pattern matching primitives; the former matches against a pattern of arbitrary length and the latter binds the value of the voltage slot of the telemetry information to a local variables named X.

```
Name:        TELEMETRY-INTEREST-FOR-
               LVDA
Data name:   TELEMETRY
Condition:   ( (! !  (VOLTAGE ?X)  ! ! )
               (<= X 11000) )
```

The associated generic rule that acts upon the acquaintance model to produce the desired behavior is:

```
(rule volunteer-info
  (if  (and  (INFO-AVAILABLE ?I)
             (ACQUAINTANCE-INTERESTED-IN
               ?I  ?ACQ) ) )
  (then  (SEND  ?I  TO  ?ACQ  AS
    UNREQUESTED-INFORMATION) ) )
```

2. *To create its network model, the LVDA needs structural information about the network. When such information is required, the PCM has to determine which acquaintance can provide it and send that agent a request.*

The LVDA knows that the IA can produce the network data for a substation because its acquaintance model indicates that it can perform a skill whose result is the NETWORK-DATA:

```
Name:     GET_NETWORK_FOR_SUBSTATION
Inputs:   SUBSTATION
Results:  NETWORK-DATA
```

The associated generic rule that produces the desired behaviour is:

```
(rule generate-service-request
  (if  (and  (INFO-NEEDED  ?I)
             (CANNOT-BE-PRODUCED-LO-
               CALLY  ?I)
             (ACQUAINTANCE-CAN-PRODUCE
               ?I  ?ACQ) ) )
  (then  (SEND  INFOR-REQUEST  TO  ?ACQ
    TO  PRODUCE  ?I) ) )
```

3. *When the LVDA receives telemetry whose operation is autoreclosure, its PCM must trigger execution of the DEAL_WITH_TELEMETRY skill.*

The trigger for the DEAL_WITH_TELEMETRY skill is defined in the Self Model. It is triggered on receipt of TELEMETRY whose voltage value is less than or equal to 11kV and whose operation is autoreclosure (the pattern-matching primitives are the same as those used in the interest descriptor).

```
Name:        TELEMETRY-TRIGGER-FOR-
               LVDA
Data name:   TELEMETRY
Condition:   (  (! !  (VOLTAGE  ?X)  ! !
                 (OPERATION  ?OP)  ! !)
               (<=  X  11000)  (OP  =  AUTO-
                 REC) )
```

The associated generic rule is:

```
(rule start-skill
  (if  (and  (RECEIVE-INFO  ?I)
             (INFO-IS-SKILL-TRIGGER ?I ?SKILL)
             (TRIGGER-SATISFIED  ?SKILL) ) )
  (then  (EXECUTE-SKILL  ?SKILL) ) )
```

5.2. Interfacing the ARCHON Layer and its Intelligent System

When the relevant parts of the ARCHON Layer have been instantiated, a connection with the IS needs to be established. In the case of purpose-built ISs, this is

a relatively straightforward activity because they are designed to be controlled from the ARCHON Layer. In this case, invoking an IS task is similar to an asynchronous function call.

In case of pre-existing systems, however, connection with the ARCHON Layer can be a substantially more demanding job. One reason for this is that the IS's control regime is embedded in the software alongside its problem-solving expertise. From a multi-agent point of view, this is unsatisfactory, the ARCHON Layer should be taking the major control decisions, since it knows about the other community members and the effects that local decisions will have on global coherence. Therefore, the IS will need to be altered to make the control more accessible and amenable to manipulation. For example, in the particle accelerator application, once the diagnosis expert systems were started, they remained in a continuous loop, and control could only be achieved by injecting items into their agenda (Jennings et al., 1993). To incorporate these systems into a cooperating community, the basic steps of the loop were identified as skills and then the top-level control loop was removed from the IS and placed in the ARCHON Layer. An example from CIDIM is that the LVES originally had its own data sources. However, when the LVDA was integrated into the cooperating community, other agents became responsible for providing the information it needed. Thus, the control of when such information was incorporated into the reasoning process was undertaken by the ARCHON Layer, rather than by the IS, where it had previously resided.

5.3. Introducing the Agents to Each Other

Once all the agents have been instantiated, they can be integrated into a community. Ideally, this final phase just involves loading the agents into the target environment, specifying the network addresses, and starting the whole system. However, before this is carried out, it is advisable to perform a few simple consistency checks on the Agent Models—especially if each agent has been developed by a different team. Pertinent checks include ensuring that interacting agents have consistent information definitions for data they are to exchange and ensuring that each agent's requests for services are capable of being executed by at least one community member.

Once the agents are plugged together, they have to be tested. This can be carried out in two ways: if the real process allows on-line experimentation, then this should be used as the test environment (as was the case with CERN's particle accelerator controller). Alternatively, a simulated environment must be developed and used as the basis for testing. In CIDIM, testing was based on simulated inputs, which were sampled from real events because the distribution network can run for long periods of time before a disturbance occurs. The purpose of this community testing phase is to ensure that each agent's individual modules have been coded correctly and also that the interactions between community members have been successfully captured.

In many applications nonfunctional requirements such as performance are an important component of the final system. For this reason, the cooperating community must meet the desired performance characteristics as well as operating correctly. Bottlenecks may occur in the computation resources of individual agents or in the communication channels between them, and this can only be detected by monitoring the performance of the working system. When such bottlenecks are detected, the designer must take appropriate steps to alleviate them. In an early version of CIDIM, for example, the TA would send all the generated telemetry to both the HVDA and the LVDA. This meant that large amounts of data were needlessly being sent to agents who simply discarded them. This problem has been overcome in the present implementation by allowing more expressive statements to be included in the conditions of interest descriptors (see Section 5.1.2). Another example is that the switch-checking agent was introduced into CIDIM to reduce a resource bottleneck. The SSPA is capable of rechecking switching plans, but if the user is in the process of creating a schedule (the SSPA's main activity), then there will be a major resource conflict. By increasing the availability of the scarce resource (the ability to recheck switching plans), CIDIM can operate more efficiently.

6. CONCLUSIONS

CIDIM can be regarded as typical of a whole class of industrial applications that are ripe for a multi-agent approach. It has many features that are often observed in this type of application: some of the ISs were pre-existing, and some needed to be purpose-built for the integrated system; there was some interaction, through the operators, before the cooperating community was implemented, although there was significant scope for improvements in the quality of such exchanges. Development of CIDIM represents an important contribution in that DAI (and AI) techniques are rarely applied to real-world problems. Consequently, many of the bottom-up design and implementation issues highlighted here remain hidden when systems are built for toy or highly idealised domains. A further contribution is in highlighting the types of cooperation that can be expected when semi-autonomous agents interact on a common problem. This aspect provides an important means of verifying the applicability of the various theoretical models of cooperation that exist within the DAI literature. In this type of application, the forms of interaction that are most beneficial are the timely sharing of relevant information and the ability to ask acquaintances to carry out tasks or provide information

that the local agent is unable to perform or provide for itself.

As a consequence of building CIDIM and several other industrial applications, a series of issues and design forces were identified as central to the process of developing real world multi-agent systems. Guidelines are given relating to the analysis of an application from a multi-agent perspective, and important design choices concerning the composition of a cooperating community are identified. At this stage, the methodology cannot be regarded as a final, neither for creating ARCHON applications nor for constructing multi-agent communities for industrial control. However, it is now sufficiently mature that still greater experience only results in slight modifications.

Future work aims to incrementally extend the CIDIM implementation. The remainder of the interactions relating to diagnosis will be coded, and work on incorporating the full functionality related to switching plans and security analysis will begin. Once a sufficiently complete cooperating community has been built, a 3-month trial will be conducted at a Regional Electricity Company to see how the system copes with on-line management of the distribution network and how it compares with the present mode of operation. This initial implementation highlighted a number of deficiencies in the ARCHON system, which need to be ironed out for the final version. In particular, run-time control of the various social and situation assessment functions needs to be improved so that agents can respond in a more timely fashion to urgent events. Second, a limited form of conflict resolution is needed, so that discrepancies can be resolved by the community at run-time rather than attempting to engineer them all out at design time, as occurs with the present implementation.

When CIDIM is complete, it will enhance the job of the CE by automating tasks that are at present extremely tedious to perform. For example rather than seeing all telemetry messages, as is the case at present, just fault diagnoses can be presented. In the worst case, this will mean being shown tens of faults in a day of which the important ones will be highlighted. This contrasts with the thousands of telemetry messages that are currently presented on an average day. Confidence in diagnoses will increase because lightning information will automatically be collated with faults. At present, this requires a time-consuming look-up, which is often not undertaken when the CE is busy. These and other automated exchanges of information will allow the CE to concentrate more on supervision of maintenance and repair work and will also allow more time to be spent on preventative actions.

In the long term, the adoption of a cooperating systems approach, coupled with other changes in the control room (such as graphical information systems), means that there will be less need to have centralised control and also that there will be greater opportunities for dynamically switching control from one centre to another. At present the latter capability is limited (unless it has been preplanned like the changeover of control at night) because all of the necessary information is not portable (e.g., a Control Centre is responsible for one region and has large wall diagrams of the network for that region). This ability to transfer control will be of use as a back-up procedure or to share the workload during severe fault conditions.

Finally, with the advances in telecommunications and radio links, it will soon be possible for the field engineer to have full access to network diagrams, safety checking, lightning data, and all the services of CIDIM whilst he is on location. Indeed, the role of the CE may change dramatically with more local decisions being taken because of the increased availability of information.

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, IRIDIA, Iberdrola, EA Technology, Amber, Technical University of Athens, FWI University of Amsterdam, CAP Volmac, CERN, and University of Porto. The ARCHON Layer concepts described in this paper have been devised and implemented by a number of partners—the Monitor by Framentec-Cognitech, the data storage mechanism by the University of Amsterdam, the Session Layer by the University of Athens, the High Level Communication Module by Atlas Elektronik and Queen Mary and Westfield College, and the Planning and Coordination Module by Queen Mary and Westfield College. The application described in this paper has been developed by EA Technology.

REFERENCES

Avouris, N. M. (1992). User interface design for DAI application. In N. M. Avouris & L. Gasser (Eds.), *Distributed artificial intelligence: Theory and praxis.* Kluwer Academic Press (pp. 141–162).

Bond, A. H., & Gasser, L., (Eds.). (1988), *Readings in distributed artificial intelligence.* Morgan Kaufmann.

Brailsford, J., Cross, A. D., & Raven, P. F. (1987). The switching schedule production assistant—a knowledge based system for the electrical power distribution engineer. *IEE Digest,* 1987/83.

Bramer, M. A., Muirden, D., Pierce, J., Platts, J. C., & Vipond, D. L. (1988). Faust—An expert system for diagnosing faults in an electricity supply system. In B. Kelly & A. Rector (Eds.). *Research and development in expert systems* 5. Cambridge University Press.

Cammarata, S., McArthur, D., & Steeb, R. (1983). Strategies of cooperation in distributed problem solving. *Proceedings of the International Joint Conference on Artificial Intelligence,* Karlsruhe, Germany (pp. 767–770).

Cockburn, D. (1992). Two model based systems for fault diagnosis in electricity distribution networks. In *IEE Digest No.* 1992/048 Colloquium on Intelligent Fault Diagnosis, Part 2: Model Based Techniques. London, UK.

Cockburn, D., McDonald, J. R., Burt, G., Brailsford, J. R., Beaton, J., & Lo, K. L. (1991). Expert systems for on-line fault diagnosis in electrical power networks. In *CIRED* 1991, Vol. 1, Session 4, pp. 4.3.1–4.3.7.

Cockburn, D., Varga, L. Z., & Jennings, N. R. (1992). Cooperating

intelligent systems for electricity distribution. *Proceedings of Expert Systems* 1992 (Applications Track), Cambridge, UK.

Corera, J., Laresgoiti, I., Cockburn, D., & Cross, A. D. (1993). A cooperative approach towards the solution of complex decision problems in energy management and electricity networks. In *12th International Conference on Electricity Distribution 1993*, **4**, 19.1–19.6.

Cross, A. D., Brailsford, J. R., & Brint, A. T. (1992). A KBS for writing safe sequences of operations on a high voltage electricity network. *First International Conference on the Practical Applications of Prolog*, **1**, 491–495.

Cross, A. D., Brailsford, J. R., & Brint, A. T. (1993). Expert systems to support network switching. In *12th International Conference on Electricity Distribution, Volume* 1. Birmingham, AL.

Erman, L. D., & Lesser, V. R. (1975). A multi-level organisation for problem solving using many diverse cooperating sources of knowledge. *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 483–490). Tblisi, Russia.

Gasser, L., & Huhns, M. N., (Eds.). (1989). *Distributed artificial intelligence, Volume II*. London: Pitman.

Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, **26**, 251–321.

Hewitt, C. E., & Kornfield, W. A. (1980). Message passing semantics. *SIGART Newsletter*, 48.

Huhns, M. N., (Ed.). (1988). *Distributed artificial intelligence*. London: Pitman.

Jennings, N. R. (1992). Using GRATE to build cooperating agents for industrial control. *Proceedings of the IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in Real Time Control*, Delft, The Netherlands, pp. 691–696.

Jennings, N. R., Mamdani, E. H., Laresgoiti, I., Perez, J., & Corera, J. (1992). GRATE: A general framework for cooperative problem solving. *Journal of Intelligent Systems Engineering*, **1**(2), 102–114.

Jennings, N. R., Varga, L. Z., Aarnts, R. P., Fuchs, J., & Skarek, P. (1993). Transforming standalone expert systems into a community of cooperating agents. *International Journal of Engineering Applications of Artificial Intelligence*, **6**(4), 317–331.

Jennings, N. R., & Wittig, T. (1992). ARCHON: Theory and Practice. In N. M. Avouris & L. Gasser (Eds.), *Distributed artificial intelligence: Theory and praxis* (pp. 179–196). Dordrecht, The Netherlands: Kluwer Academic Press.

Klein, M. (1991). Supporting conflict resolution in cooperative design systems. *IEEE Transactions on Systems, Man and Cybernetics*, **21**(6), 1379–1390.

Lees, M. I. (1992). Measurement of lightning ground strikes in the UK. *Lightning Protection*, 92.

Lesser, V. R., & Erman, L. D. (1980). An experiment in distributed interpretation. *IEEE Transactions on Computers*, **29**(12), 1144–1163.

Nii, P. H. (1986a). Blackboard systems (Part I). *AI Magazine*, **7**(2), 38–53.

Nii, P. H. (1986b). Blackboard systems (Part II). *AI Magazine*, **7**(3), 82–106.

Oliveira, E., Camacho, R., & Ramos, C. (1991). A multi-agent environment in robotics. *Robotica*, **4**(9).

Parunak, H. V. D. (1987). Manufacturing experience with the contract net. In M. N. Huhns (Ed.). *Distributed Artificial Intelligence* (pp. 285–310). London: Pitman.

Reddy, Y. V., Srinivas, K., Jagannathan, V., & Karinthi, R. (1993). Computer support for concurrent engineering. *IEEE Computer*, **26**(1), 12–17.

Roda, C., Jennings, N. R., & Mamdani, E. H. (1991). The impact of heterogeneity on cooperating agents. *Proceedings of the AAAI workshop on cooperation among heterogeneous intelligent systems*, Anaheim, CA.

Scott, L. J. (1988). A lightning location system for the UK electricity supply industry. *International Conference on Lightning and Static Electricity* (pp. 20–23). Birmingham, AL.

Stassinopoulos, G., & Lembessis, E. (1993). Application of a multi-agent cooperative architecture to process control in the cement factory (ARCHON Technical Report 43/3-93).

Weihmayer, R., & Brandau, R. (1990). Cooperative distributed problem solving for communication network management. *Computer Communications*, **13**(9), 547–557.

Wittig, T. (Ed.). (1992). *ARCHON: An architecture for multi-agent systems*. Sussex, UK: Ellis Horwood.