Agent Software

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

1. Intelligent Agents and Multi-Agent Systems

The term "*agent*", (and hence "agent based computing", "agent based system", "multi-agent system"), is being increasingly used within information technology to describe a broad range of computational entities. These entities range from relatively simple systems, (such as Microsoft's TIP WIZARD, which provides advice to users working in EXCEL 5, or desktop agents which prioritise and filter electronic mail (Maes, 1994)), right up to very large, interoperable expert systems or databases which contain thousands of lines of code (e.g., ARCHON (Jennings *et al*., 1995) and Carnot (Huhns *et al*., 1992)). Broadly speaking, we can identify three different classes of agent. At the simplest level, there are "*gopher*" agents, which can execute straightforward tasks based on pre-specified rules and assumptions (e.g., remind me that I have a lecture every Monday this term at 2:00). The next level of sophistication involves "*service performing*" agents, which execute a well-defined high-level task at the request of a user (e.g., arrange a meeting or find an appropriate flight). Finally, there are "*predictive/proactive*" agents, which volunteer information or services to a user, without being asked, whenever it is deemed to be appropriate (e.g., an agent may monitor news groups on the Internet and return discussions that it believes to be of interest to the user).

Given this obvious diversity, it is important to firstly identify the characteristics which are common to these systems and which enable their developers to meaningfully bestow the tag of agenthood upon them. As with many definitions in our field, what follows is not universally accepted - however most researchers will probably find themselves in broad agreement with the majority of its sentiments. An agent is a self contained problem solving entity (implemented in hardware, software or a mixture of the two) which exhibits the following properties (Wooldridge and Jennings, 1995):

- *Autonomy*: agents should be able to perform the majority of their problem solving tasks without the direct intervention of humans or other agents, and they should have a degree of control over their own actions and their own internal state.
- *Social ability*: agents should be able to interact, when they deem appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities. This requires that agents have, as a minimum, a means by which they can communicate their requirements to others and an internal mechanism for deciding when social interactions are appropriate (both in terms of generating appropriate requests and judging incoming requests).

- *Responsiveness*: agents should perceive their environment (which may be the physical world, a user, a collection of agents, the INTERNET, etc.) and respond in a timely fashion to changes which occur in it.

- *Proactiveness*: agents should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where it is appropriate.

In addition to these necessary conditions, a number of other potentially desirable characteristics have been proposed (Wooldridge and Jennings, 1995). These include: *adaptability* - the ability of an agent to modify its behaviour over time in response to changing environmental conditions or an increase in knowledge about its problem solving role; *mobility* - the ability of an agent to change its physical location to enhance its problem solving; *veracity* - the assumption that an agent will not knowingly communicate false information; and *rationality* - the assumption that an agent will act in order to achieve its goals and will not act in such a way as to prevent its goals being achieved without good cause.

Having defined some key characteristics of an agent, the next step is to identify why, in various quarters, agent based computing has been hailed as "the next significant breakthrough in software development" (Sargent, 1992) and "the new revolution in software" (Ovum, 1994). Indeed according to one UK market researcher, the worldwide market for agent software will grow from an annual value of $37 million in 1994 to $2.6 billion by the year 2000.

For any new technology to be considered as having such a large potential in the computer marketplace, it must offer one of two things: (i) the ability to solve problems which have hitherto been beyond the scope of automation - either because no technology could be used to solve the problem (very rare) or because it was considered too expensive to develop solutions using the previously available technology (more likely); or (ii) the ability to solve problems which can already be solved in a better (more natural, easy, efficient, or fast) way.

With respect to the former motivation, previously untackled problems for which agent based systems are now being developed, can be characterised as having the following properties:

- Openness

  Components of the system are not known in advance, can change over time, and are highly heterogeneous (in that they are implemented by different people, at different times, using different problem solving paradigms). Computing applications are increasingly demanded by users to operate in such domains. Perhaps the best known example of a highly open software environment is the Internet, a loosely coupled computer network of ever expanding size and complexity. The design and construction of software tools to exploit the enormous potential of the Internet and its related technology is one of the most important challenges facing computer scientists in the 1990s. In domains such as the Internet, an agent based approach is needed because the components with which an individual must interact are not known *a priori*. Thus the agents and their interfaces to one another must be flexible and robust because few of the interactions can be predicted at design time. Agents must be endowed with sophisticated social skills such as persuasion abilities to make others come around to their point of view (Sycara, 1989), negotiation abilities to help reach agreements (Müller, 1995), and coordination mechanisms (Durfee, 1988) to describe how joint or interrelated activities should be performed. All of these social activities

need to be underpinned by a communication mechanism which enables the various objectives to be transmitted in a mutually comprehensible manner. Openness also requires the agents to continually monitor their environment and their objectives (since, in general, they cannot predict very far in advance what is likely to happen next) and to adopt new goals proactively when the appropriate opportunities arise.

- Complexity

  The domain is so large, sophisticated, or unpredictable that the only way it can reasonably be addressed is to develop a number of (nearly) modular components which are specialised (in terms of their representation and problem solving paradigm) at solving a particular aspect of it. In such cases, when interdependent problems arise the agents have to interact with one another to ensure their dependencies are properly managed. Real world examples of such domains for which agent based systems have been developed include: power systems management (Jennings *et al*., 1995; Varga *et al*., 1994), particle accelerator control (Jennings *et al*., 1993), telecommunications network management (Weihmayer and Velthuijsen, 1994), spacecraft control (Schwuttke and Quan, 1993), computer integrated manufacturing (Parunak, 1995), transportation management (Fischer *et al*., 1995), job shop scheduling (Morley and Schelberg, 1993) and steel coil processing (Mori *et al*., 1988).

  In such domains, an agent based approach means that the overall problem can indeed be partitioned into a number of smaller and simpler components, which are easier to develop and maintain, and which are specialised at solving the constituent subproblems in an effective manner. This decomposition allows each agent to employ the most appropriate paradigm for solving its particular problem, rather than being forced to adopt a common uniform approach which represents a compromise for the entire system but which is not optimal for any of its subparts. Agent based systems represent something more than the classical divide and conquer methodology of traditional software engineering, in that the agents must be able to interact in a flexible, context dependent manner (hence the need for social abilities, responsiveness, and proactiveness) rather than through some fixed and predetermined set of interface functions. Also, the unpredictability of the domain means that the agents must be both responsive to change and proactive, in that if unexpected opportunities arise then they must be able to opportunistically grasp them. See (Jennings, 1994) for a detailed discussion of the issues involved in developing agent based systems for complex industrial applications.

The second stated reason for adopting a new technology is that it provides a better, along some dimension, means of conceptualising and/or implementing a given application. Here there are three important domain characteristics which are often cited as a rationale for adopting agent technology (cf. (Bond and Gasser, 1988)):

- Distribution of data, control, expertise or resources

  When the domain involves a number of distinct problem solving entities (or data sources) which are physically or logically distributed (in terms of their data, control, expertise or resources) and which need to interact with one another (or be combined) in order to solve their problems (or one common problem), then agents can often provide an effective solution. For example, in a distributed health care setting general practitioners, hospital specialists, nurses, and home care organisations have to work

together to provide the appropriate care to a sick patient (Huang *et al*., 1995). In this scenario, which is well suited to an agent based solution, there is a distribution of: data (the general practitioner has his own data about the patient which is very different from that of the hospital nurse even though it concerns the same person), control (each individual is responsible for performing a different set of tasks), expertise (the specialist's knowledge is very different from that of either the general practitioner or the nurse), and resources (a specialist is responsible for the beds which his patients need, a general practitioner for paying for the hospital services, etc.).

In such cases, an agent based solution provides a natural means of modelling the problem - real world entities and their interactions can be directly mapped into autonomous problem solving agents which have their own resources and expertise and which are able to interact with others in order to get tasks done. Also, in the case of distributed data sources, (as in sensor networks (Lesser and Corkill, 1983) and seismic monitoring (Mason, 1995)), the use of agents means that significant amounts of processing can be carried out at the data's source, with only high-level information exchanged. This alleviates the need to send large amounts of raw data to a distant central processor, thus making more efficient use of communications bandwidth.

- Natural Metaphor

  The notion of an autonomous agent can be the easiest way of conceptualising or presenting a given software functionality. For example, a program which filters email can be presented to its user via the metaphor of a personal digital assistant (Maes, 1994), meeting scheduling software can naturally be presented as an empowered, autonomous, social agent which can interact with other similar agents on the user's behalf (Jennings and Jackson, 1995). In such applications the fact that these functions are implemented through a series of local agents also means that they can be personalised to reflect the preferences of their user. Also in computer games (Wavish and Graham, 1995) and virtually reality systems (Bates, 1994), characters can naturally be represented as self-contained, autonomous, social problem solving entities (i.e., agents).

- Legacy Systems

  In many traditional areas of computing (such as computer integrated manufacturing (Parunak, 1995) and process control (Jennings and Wittig, 1992)), there is a significant amount of existing software (especially information systems) which performs critical organisational functions. To keep pace with changing business needs, this software must be periodically updated. However, modifying such legacy systems is in general very difficult, since the system's structure and detailed working will invariably have become corrupted with the passage of time, and a complete rewrite would be prohibitively expensive (if it were even possible!). Therefore, in the long term, the only way such legacy systems can remain useful is to incorporate them into a wider cooperating community (a cooperative information system (Papazoglou *et al*., 1992)) in which they can be exploited by other pieces of software. This can be done, for example, by building an "agent wrapper" around the software to enable it to interoperate with other systems (Genesereth and Ketchpel, 1994; Jennings *et al*., 1993).

Although agent based technology clearly has an important role to play in the development of leading edge computing applications, it should not be regarded as a panacea. The majority of applications which currently use agents could be solved using non-agent techniques (in most cases not as well, but in some cases better!). Thus the mere fact that a particular problem domain is open or involves legacy systems does not necessarily imply that an agent based solution is the best one (or even that it is a feasible one). As with all system designs, the ultimate choice depends upon a large number of technical and non-technical factors - what this section has identified is the types of situation in which an agent based solution should be considered, as opposed to those in which it should definitely be deployed.

2. Agent Applications: Two Case Studies

Agent applications herald a fundamentally new paradigm for developing and implementing complex systems. The traditional (idealised) software engineering model of providing a complete system specification and then implementing it in a number of rigid, deterministic components (modules) is inappropriate for the types of application for which agents are being considered. The agent based system development paradigm involves building sophisticated, self-contained components, which can interact flexibly with a number of independently developed similar components. Interaction can no longer be via a rigid and predetermined interface, rather it has to be achieved through negotiation and persuasion and followed up by commitments and agreements between the parties involved. This empowerment of the individual components means that the system's global properties and behaviour cannot be preprogrammed, rather they emerge out of the actions and interactions of the constituent agents at run time.

Whilst this new system paradigm offers many exciting opportunities, it has a down side which invariably places a limit on the types of application to which agents can be applied. The first major problem is that the overall system is unpredictable and non-deterministic: which agents will interact with which others in which ways to achieve what cannot be predicted in advance. Even worse, there is no guarantee that dependencies between the agents can be managed effectively, since the agents are autonomous and free to make their own decisions. Thus indefinite deadlock and starvation may occur. The second main disadvantage is that the behaviour and properties of the overall system cannot be fixed at design time. While a specification of the behaviour of an individual agent can be given, a corresponding specification of the system in its entirety cannot, since global behaviour necessarily emerges at run time.

By means of an illustration of the types of agent systems which are *currently* being developed and deployed, the remainder of this section gives a brief overview of an agent based electricity distribution management application (section 2.1) and an agent based health care management application (section 2.2).

2.1 Agent Based Electricity Distribution Management

This application, called CIDIM (Cockburn and Jennings, 1995), was developed using the ARCHON framework (Jennings *et al.*, 1995). CIDIM is being developed as an aid for control engineers who are 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 standalone 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).

Based on this description, CIDIM's main top level goals can be identified:

- 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 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 interrelations between the tasks and the data (e.g. fault diagnosis requires telemetry and network data).

When developing CIDIM, it was important that the pre-existing and standalone systems which were already used in the Control Room could be incorporated - these included:

- High Voltage Expert System (HVES): 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): 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: 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.

By comparing the high level requirements 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 agents 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 running multi-agent system although they do exist in their non-integrated form.

With the development of these new agents, the overall system provides 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 agents. 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 domain part 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 domain part of the IA when CIDIM is ported to a different company.

It was decided to provide a common user interface to this application 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 1.

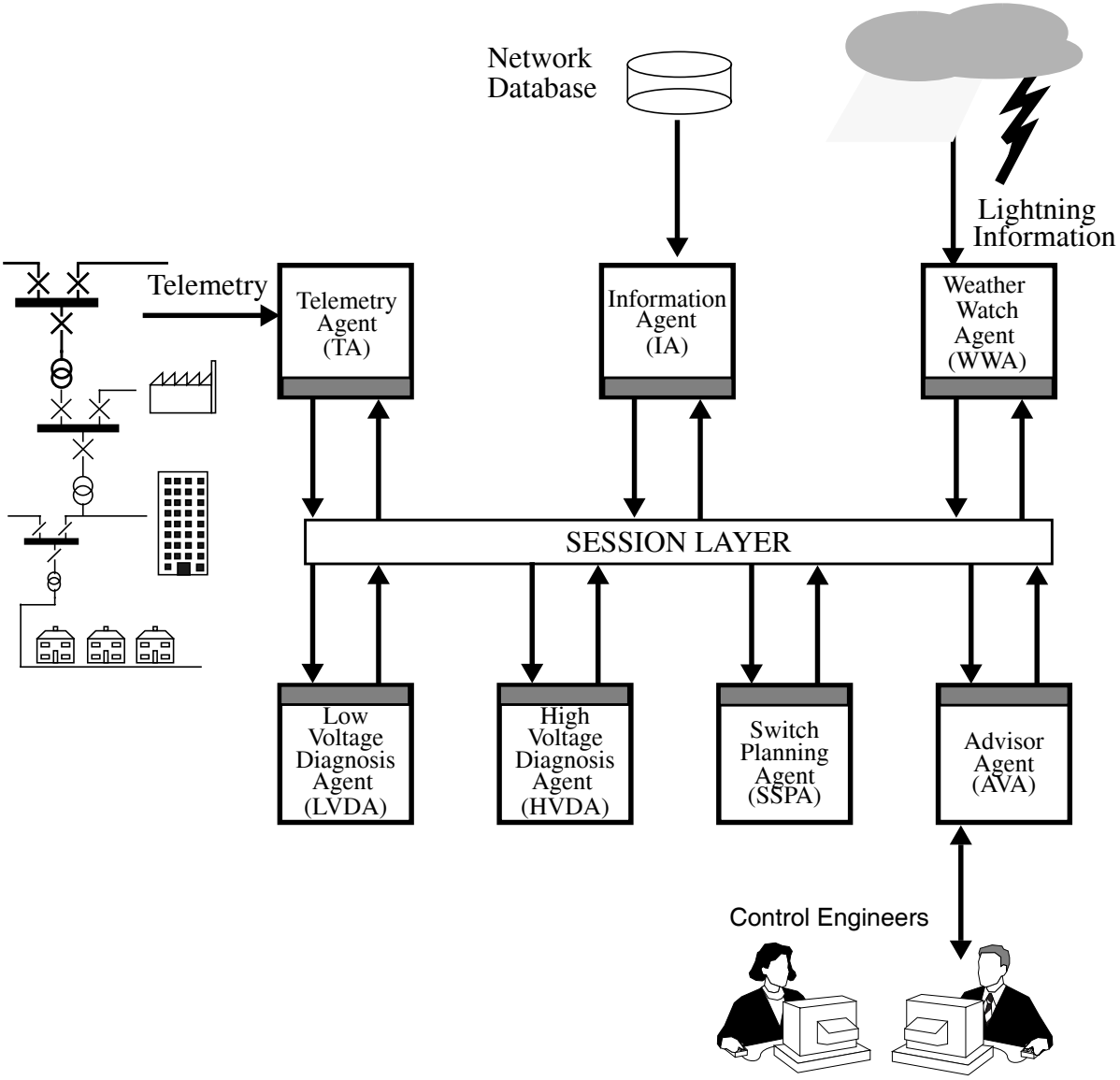

Figure 1: CIDIM's Cooperating Agents

Integrating these different subsystems 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.

2.2 Agent Based Health Care Management

Artificial Intelligence and knowledge based systems are assuming an increasingly important role in medicine for assisting clinical staff in making decisions under uncertainty (eg diagnosis decisions, therapy and test selection, and drug prescribing). Furthermore, many medical procedures now involve several individuals, in a number of specialist institutions (or departments), whose decisions and actions need to be coordinated if the care is to be effective and efficient. For example, a general practitioner (GP) may suspect that his patient has breast cancer. However, as he neither has the knowledge nor the resources to confirm this hypothesis, he must refer the patient to a hospital specialist who can make a firm diagnosis. Having confirmed the presence of breast cancer, the specialist must devise a care programme for treating the patient - this typically involves the hospital, the patient's GP, and a home care organisation jointly executing a series of interrelated tasks. In addition to this inter-organisation coordination, there is also a need to ensure that the activities within an organisation are effectively and coherently managed. In a hospital, for instance, care typically involves execution of interrelated tasks by doctors, nurses, pharmacy, laboratories, and resource management departments.

To provide the appropriate software support for such coordinated health care management it was decided to adopt an agent based approach. This decision was based on three main observations about the medical care management domain (given below) and the properties of autonomy, social ability, reactivity and proactiveness which are normally associated with intelligent agents (see section 1). The first relevant domain property is the fact that there is a significant physical distribution of information, problem-solving capabilities, resources, and responsibilities which need to be brought together in a consistent and coherent fashion by the distributed 'agents' who jointly execute a care programme (here agent is defined as an integrated entity involving a computer system and its user). Secondly, the combination of the

aforementioned decentralisation and the high cost of obtaining a comprehensive (complete) overview means that decisions often have to be made with incomplete information (eg diagnosis may be proposed without exhaustive laboratory investigation). Finally, as the environment is dynamic and unpredictable the problem solvers need to exhibit intelligent goal-oriented behaviour yet still be responsive to changes in their circumstances - plans to achieve particular goals need to be devised and whilst these plans are being executed they need to be continuously monitored (and perhaps refined) in the light of changes in information and problem solving state.

Given these domain properties and previous experience with medical care management systems, the essential features of an agent based system for this application area can be defined. Firstly, the agents need explicit communication management procedures (dealing with both syntax and semantics) so that the sender and receiver of a message have a common understanding of its meaning and purpose (in non-automated systems, human to human messages were often misinterpreted during extensive interactions because of ambiguities in the communication structures). Secondly, appropriate mechanisms and structures are needed to ensure that tasks are delegated to the most appropriate agents (previously tasks were allocated to the wrong agents and thus delays in the delivery of care occurred - a serious concern as time is such a critical factor in care administration). Thirdly, the agents require a decision making mechanism which is able to reason with contradictory and incomplete information (previously the popularly used decision methods, especially those based on probabilistic theory, could not tolerate conflicting or incomplete information). Finally, to ensure coherent care in spite of the dynamic and unpredictable environment the agents need to specify and adopt an explicit set of procedures for monitoring their goals and plans (previously no explicit procedures existed and changes in goals and care plans were managed largely in an *ad-hoc* and ineffective manner). The detailed design of the individual agents and of the entire system are given in Huang *et al.* (1995).

To illustrate the role and functionality of the agents in this application, consider the following implemented scenario. When an oncologist in a cancer hospital has to treat a patient's breast cancer, the first decision he has to make concerns the treatment plan which will be adopted. To assist him in making this decision, the oncologist consults one of his decision support systems. This system has a built-in decision procedure which is able to deal with incomplete or intuitively inconsistent information, such as evidence in favour of a choice and evidence against the choice. Having weighted the pros and cons of using various treatment options, the system recommends the use of a particular chemotherapy protocol called 'CT1 protocol'. The oncologist authorises use of this protocol and requests the computer system to support him in carrying out the treatment.

The CT1 protocol consists of a number of treatment stages, one of which, stage 2, is shown in Figure 2. According to the protocol, stage 2 is decomposed into a sequence of three subtasks: 'admit patient to hospital', 'administer drugs and monitor patient' and 'discharge patient'. The support system recommends that the oncologist carries out the first task because it knows that he is formally responsible for the admission of his patients. This recommendation is endorsed by the oncologist and consequently he takes on the role of managing and actually performing the activity. On the recommendation of his support system, the oncologist then decomposes 'admit patient to hospital' into two parallel subtasks: 'allocate bed' and 'obtain patient consent'. The machine recommends, and the oncologist accepts, that 'allocate bed' should be performed by the hospital's resource management department and 'obtain patient consent' should be carried out by the oncologist. With respect to the former subtask, the oncologist

sends an electronic request to the resource department to see whether they are willing to take on the responsibility for performing it. Assuming they are and that the task is successfully completed, the patient will be allocated a bed. Once a bed is available and the patient agrees to be admitted to the hospital, the support system recommends that the task 'administer drugs and monitor patient' is allocated to a hospital nurse. Assuming she accepts, the protocol dictates that the task should be decomposed into the sequential subtasks of 'obtain drug', 'administer drug' and 'observe patient' - all of which the nurse takes responsibility for. She may subsequently decompose the 'observe patient' subtask still further: for example, into 'measure body temperature', 'take blood samples', and 'analyse blood samples' and this decomposition may well involve generating a request to a laboratory to test patient indicators, such as white blood cell count. Finally, the machine recommends that the third subtask of stage 2, 'discharge patient', and its two subtasks, 'Instruct Patient' and 'Inform GP' should be carried out by the oncologist.

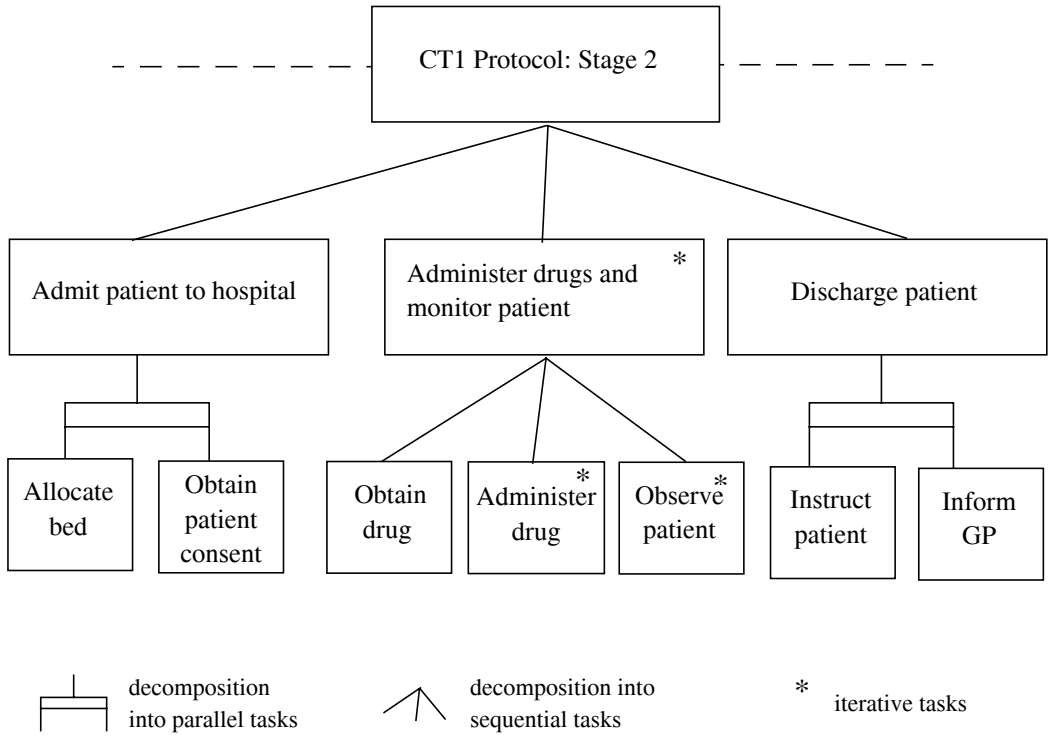

Figure 2: Part of CT1 protocol for treating breast cancer

In this scenario, agents exchange information and responsibilities according to the following pattern: the resource management department must inform the oncologist about the outcome of 'allocate bed' (i.e. either that a bed has been allocated as requested or that no bed is available for the requested date) and the nurse has to inform the oncologist of the results of 'administer drugs and monitor patient' (e.g. drug has been administered and all patient indicators are normal). Accompanying this information exchange is a concomitant flow of control, in terms of commitments and expectations (Jennings, 1993), between the agents: the oncologist expects the resource management department to perform the activity 'allocate bed' once it has agreed to, similarly he expects the nurse to perform the 'administer drugs and monitor patient' task on time once she has consented to execute it. Again the detailed design behind these functionalities are described in Huang *et al.* (1995).

3. Future Challenges and Open Problems

Although a number of agent based systems have now been deployed, and a much greater number of advanced prototypes for real world problems have been built, many open problems and challenges still remain. In terms of system design, the major issues include:

- The development of a methodology for designing agents and agent based systems

  Surprising little work has been undertaken on methodological aspects of agent based systems, and yet if this technology is to be a commercial success then designers must have a structured way of developing well-engineered agents and agent systems. This work needs to identify how robust and flexible individual agents can best be designed and how these well designed components can then be combined to give an overall system which is similarly robust and flexible. The problem of providing a system level description is made more difficult by the fact that many of its properties can only be observed at runtime. Note that some attempts have been made to investigate formal methods for agent based systems (Wooldridge, 1995). This preliminary work has been successful in identifying the key issues in the formal specification and verification of agent based systems, and a number of simple demonstrator applications and multi-agent languages have been specified. However, as is the case in mainstream computer science, the complexity of such methods means that their use is not likely to become commonplace in the immediate future.

- The development of benchmarks for evaluating design options

  At present, there is little empirical data that can be used to systematically evaluate or compare the performance of different agent designs or different means of providing various agent functionalities (see Decker and Lesser (1993) for a notable exception). This means the agent designer has little "conventional wisdom" about what mechanisms work well in which situations on which he can base his design. What would lead to better informed agent designs is a series of benchmark performance tests on which competing alternatives could be objectively compared and contrasted.

- The development of reusable tools

  By their very nature, agent based systems require a high level of basic infrastructure to be in place before they can operate effectively. Examples of such components include: communication mechanisms and protocols for message interchange, the ability to interoperate over heterogeneous platforms and debugging/monitoring facilities. In the majority of cases, this infrastructure has to be implemented from scratch for each new application. However, far greater long term productivity could be attained if these development and debugging facilities were available as a suite of reusable tools, since it would free the designer to concentrate on the more advanced agent level features.

In terms of the design of the individual agents, the following major issues still need to be addressed in a more adequate manner if agent technology is to reach its full potential:

- Heterogeneity

  Agent based research is only just beginning to grapple with problems associated with the inevitable heterogeneity of its problem solving components. The basic problem is

how agents with different domains of discourse, employing different knowledge representation schemes, different problem solving paradigms, and with different assumptions about their world and each other, can be made to interact in an effective and scalable manner.

- Reasoning with uncertain, incomplete and contradictory information

  As agents have a necessarily partial perspective of their world, and because their problem domain is open, complex and distributed, they require sophisticated mechanisms for reasoning with uncertain, incomplete and contradictory information if they are to exhibit the desired degree of flexibility and robustness. In addition to the vagueness associated with their domain, agents also have to contend with uncertainty in their interactions (e.g., "will agent A do action B for me on time?" and "should I assume that agent C will provide information I on time?") and in the information which ensues from these interactions (e.g., how should an agent treat information it receives from another agent - as highly believable or with great mistrust?, and how should it act if the information conflicts with or corroborates its own findings?). Central problems in this work include: (i) the means by which heterogeneous agents can accurately convey the semantics of uncertainty to one another (for example, a certainty measure of 0 may represent indifference to one agent and complete disbelief to another); (ii) the mechanisms which should be employed to make assumptions about the actions of others or about missing information; and (iii) the means by which the root cause of conflicts can be ascertained and, if necessary, the means by which they can be removed.

- Real time operation

  As agents are being increasingly used in time critical domains, it is important that they can give provably real-time responses in key situations. In addition to the mechanisms for delivering real time behaviour within an agent, work is also needed to make social interactions between autonomous agents more predictable - for instance, by providing a means of shortening a protracted negotiation dialogue or a means by which one agent can exert sufficient influence over another to get what it considers to be an urgent task processed with a higher priority.

- Adaptability

  In the types of environment in which agents are typically situated, it is important that they can acquire more information about their surroundings and about their problem solving role at run time (since, in many cases, it will be impossible to do it at design time). Having attained this increased awareness, the agent must then be able to adapt its behaviour so that it can perform its problem solving more effectively.

Acknowledgements

This paper describes joint work which has been undertaken with a number of other people and I am happy to acknowledge their contribution. The introductory material on agents was developed jointly with Mike Wooldridge from Manchester Metropolitan University. The

electricity distribution management work was carried out in the ESPRIT II project ARCHON (P-2256) whose partners were Atlas Elektronik, Framentec-Cognitech, Labein, Queen Mary and Westfield College, Iberdrola, EA Technology, Iridia, Amber, Technical University of Athens, FWI University of Amsterdam, CAP Volmac, CERN and University of Porto. Individuals who made particular contributions to this project include: architecture design (Thies Wittig, Abe Mamdani, Erick Gaussens), the Monitor (Erick Gaussens, Daniel Gureghian, Jean-Marc Loingtier, Bernard Burg), the PCM (Nick Jennings, Jeff Pople, Jochen Ehlers, Eugenio Oliveira), AIM (Frank Tuijnman, Hamideh Afsarmanesh, Giel Wiedijk), the HLCM (Claudia Roda, Jutta Müller), the Agent Models (Nick Jennings) and the C++ implementation (Rob Aarnts). The Electricity Distribution application was developed jointly with EA Technology and in particular with David Cockburn and Andrew Cross. The health care application represents joint work with John Fox from the Imperial Research Cancer Fund and Jun Huang from the University of Central Lancashire.

REFERENCES

J. Bates. 1994. The role of emotion in believable agents. Comms. of the ACM 37 (7) pp 122-125.

A. H. Bond and L. Gasser, eds. 1988. Readings in Distributed Artificial Intelligence. Morgan Kaufmann.

D. Cockburn and N. R. Jennings. 1995. ARCHON: A Distributed Artificial Intelligence System for Industrial Applications. in Foundations of Distributed Artificial Intelligence (eds. G. M. P. O'Hare and N. R. Jennings) Wiley.

K. S. Decker and V. R. Lesser. 1993. Quantitative Modelling of Complex environments. Int. Journal of Intelligent Systems in Accounting Finance and management 2 (4) pp 215-234.

E. H. Durfee. 1988. Coordination of distributed problem solvers. Kluwer Academic Publishers.

K. Fischer, J. P. Müller and M. Pischel. 1995. Cooperative transportation scheduling: an application domain for DAI. Journal of Applied Artificial Intelligence (Special Issue on Intelligent Agents - eds M. J. Wooldridge & N. R. Jennings).

M. R. Genesereth and S. P. Ketchpel. 1994. Software agents. Comms. of the ACM 37 (7) pp 48-53.

J. Huang, N. R. Jennings, and J. Fox. 1995. An agent based approach to health care management. Journal of Applied Artificial Intelligence (Special Issue on Intelligent Agents - eds M. J. Wooldridge & N. R. Jennings).

M. N. Huhns, N. Jacobs, T. Ksiezyk, W. M. Shen, M. P. Singh, and P. E. Cannata. 1992. Integrating enterprise information models in Carnot. Proc. Int. Conf. on Intelligent and Cooperative Information Systems. Rotterdam, The Netherlands, pp 32-42

N. R. Jennings. 1994. Cooperation in industrial multi-agent systems. Series in Computer Science, Vol 43, World Scientific Publishing, Singapore.

N. R. Jennings. 1993. Commitments and Conventions: The Foundation of Coordination in

Multi-Agent Systems. The Knowledge Engineering Review, 8 (3) 223-250.

N. R. Jennings, J. Corera, I. Laresgoiti, E. H. Mamdani, F. Perriolat, P. Skarek and L. Z. Varga. 1995. Using ARCHON to develop real-word DAI applications for electricity transportation management and particle accelerator control. IEEE Expert - Special Issue on Real World Applications of DAI.

N. R. Jennings and A. J. Jackson. 1995. Agent based Meeting Scheduling: A Design and Implementation. IEE Electronics Letters Journal.

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

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

V. R. Lesser and D. D. Corkill. 1983. The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving network. AI Magazine, Fall, pp 15-33.

P. Maes. 1994. Agents that reduce work and information overload. Comms. of the ACM 37 (7) pp 31-40.

C. L. Mason. 1995. Cooperative interpretation of seismic data for nuclear test ban treaty verification: a DAI approach. Journal of Applied Artificial Intelligence (Special Issue on Intelligent Agents - eds M. J. Wooldridge & N. R. Jennings).

K. Mori, H. Torikoshi, K. Nakai, K. Mori, and T. Masuda. 1988. Computer control system for iron and steel plants. Hitachi Review 37 (4) pp 251-258.

R. E. Morley and C. Schelberg. 1993. An analysis of a plant-specific dynamic scheduler. Proc. of the NSF Workshop on Dynamic Scheduling, Cocoa Beach, Florida.

H. J. Müller. 1995. Negotiation principles. In Foundations of Distributed Artificial Intelligence (eds G. M. P. O'Hare and N. R. Jennings) Wiley.

Ovum Report. 1994. Intelligent agents: the new revolution in software.

M. P. Papazoglou, S. C. Laufman, and T. K. Sellis. 1992. An organisational framework for cooperating intelligent information systems. Journal of Intelligent and Cooperative Information Systems. 1 (1) pp 169-202.

H. V. D. Parunak. 1995. Applications of distributed artificial intelligence in industry. In Foundations of Distributed Artificial Intelligence (eds. G. M. P. O'Hare and N. R. Jennings), Wiley.

P. Sargent. 1992. Back to school for a brand new ABC. In The Guardian, 12 March, p 28.

U. M. Schwuttke and A. G. Quan. 1993. Enhancing performance of cooperating agents in real time diagnosis systems. Proc. 13th Int. Joint Conference on Artificial Intelligence, Chamberry, France, pp 332-337.

K. P. Sycara. 1989. Argumentation: planning other agents plans. Proc. Int. Joint Conf. on AI, Detroit, Michigan, pp 517-523.

L. Z. Varga, N. R. Jennings and D. Cockburn. 1994. Integrating intelligent systems into a cooperating community for electricity distribution management. Int Journal of Expert Systems with Applications 7 (4) pp 563-579.

P. Wavish and M. Graham. 1995. A situated action approach to implementing characters in computer games. Journal of Applied Artificial Intelligence (Special Issue on Intelligent Agents - eds M. J. Wooldridge & N. R. Jennings).

R. Weihmayer and H. Velthuijsen. 1994. Application of distributed AI and cooperative problem solving to telecommunications. In AI Approaches to Telecommunications and Network Management (eds J. Liebowitz and D. Prereau) IOS Press.

M. Wooldridge. 1995. This is MyWorld: The Logic of an Agent-Oriented DAI testbed. In Intelligent Agents - Theories, Architectures, and Languages (eds. M. Wooldridge and N. R. Jennings). Springer-Verlag Lecture Notes in Artificial Intelligence Volume 890, pp 263--274.

M. Wooldridge and N. R. Jennings. 1995. Intelligent agents: theory and practice. Knowledge Engineering Review 10 (2).