Software Agents: A review

27 May 1997

Trinity College Dublin[1]

Shaw Green
Leon Hurst
Brenda Nangle
Dr. Pádraig Cunningham

Broadcom Éireann Research Ltd.[2]

Fergal Somers
Dr. Richard Evans

[1] {shaw.green, leon.hurst, brenda.nangle, padraig.cunningham}@cs.tcd.ie
[2] {fs, re}@broadcom.ie

Table of Contents

1. Introduction

In September 1996, Broadcom Ireland formed a research collaboration with the Computer Science Department in Trinity College Dublin., in order to explore current research in the domain of Intelligent Agents and to apply this technology to applications in communications. The resulting collaboration has been named the ***Intelligent Agents Group (IAG)***, consisting of four members from TCD working in liaison with a group of similar size from Broadcom.

Intelligent Agents are one of the "hot" topics in Information Systems R&D at the moment. The last ten years have seen a marked interest in agent-oriented technology, spanning applications as diverse as information retrieval, user interface design and network management. This diversity means that the IA field presents a very confusing picture. The main goal of this IAG review is to report on research in the rapidly evolving area of software agents and to highlight the applicability of this technology to our sponsor's interests.

Chapter 2 introduces the concept of an agent, the reasons for the increasingly growing interest in this field and briefly outlines the key aspects of the three areas the IAG has chosen for research, namely, Intelligent User Interfaces, Distributed Agent Technology and Mobile Agent Technology. Chapters 3, 4 and 5 will then provide in-depth overviews on each of these areas, outlining the underlying technologies and associated research projects and existing applications. Finally, section 6 will conclude this review with an overall summary.

More information is available from our WWW pages at: http://www.cs.tcd.ie/Brenda.Nangle/iag.html

2. Agent technology: an overview

2.1 Introduction

The word 'agent' is currently in vogue in the popular computing press and within the artificial intelligence and computer science communities. This chapter provides the reader with a brief overview of what an agent is and of three key agent application areas. In section 2.2, we will provide a general agent definition and outline the reasons for the increasing interest in agent technology. Sections 2.3, 2.4 and 2.5 will briefly summarise the key aspects associated with the three areas under research by the Intelligent Agents Group.

2.2 General Agent definition

We are all in one sense familiar with the concept on agent. Consider for instance the role undertaken by a travel or estate agent. Both these organisations have a representative role, that is they both ***act on behalf of others***, in the case of the estate agents this is the actual owners of the property, for travel agents the hotels and flight companies. We claim acting on behalf of another entity is the first fundamental property of agency. The agents discussed above also exhibit a second fundamental agent characteristic, namely, they both enjoy at least a variable degree of ***autonomy***. For example, estate agents can generally make viewing appointments for unoccupied properties without reference to the owners. A third important aspect of an agent's behaviour is the degree of ***proactivity***[3] and ***reactivity***[4] present in their behaviour. For instance, an estate agent who simply places a 'For Sale' sign outside a property for sale and waits for purchasers to come into his shop is behaving in a much more reactive fashion, than an agent who proactively advertises the property in the local press. It should be noted however that reactivity and proactivity are not flip sides of the same coin. The same agent can display high amounts of both proactivity and reactivity at different times. Finally, agents also exhibit some level of a number of attributes, the key ones of which are ***learning, cooperation and mobility***.

The above discussion is equally true of *computational agents* in general. The term 'agent' in this context has its background in the early work on AI when researchers concentrated on trying to create artificial entities which mimicked human abilities [Hewitt, 1977]. In its strictest sense, the term can be applied to a wide range of entities including the software systems which have become synonymous with the term as well as autonomous robots and biological organisms. In essence, therefore, we believe that the concept of agenthood can be summed up by the following definition

" An agent is a computational entity which :-
- *acts on behalf of other entities in an autonomous fashion*
- *performs its actions with some level of proactivity and/or reactiveness*
- *exhibits some level of the key attributes of learning, co-operation and mobility".*

Software agents (often simply termed agents) are software systems that loosely conform to the above definition and can basically be described as inhabiting computers and networks, assisting users with computer-based tasks. Why is it that in the closing years of the 20th century the need for software agents is becoming so urgent ? In today's modern world, computers are now as ubiquitous as cars or televisions, but exploiting their capabilities is a problem we still need to tackle. As this technology proliferates, the gap between millions of untrained users and an equal number of sophisticated microprocessors will become even more apparent. Computers currently respond only to what interface designers call direct manipulation. Nothing happens unless a person gives commands from a keyboard, mouse or touch screen. The computer is merely a passive entity waiting to execute specific, highly detailed instructions; it provides little help for complex tasks or for carrying out actions that may consume a large proportion of the user's time (e.g., searching for information). Researchers and software companies have set high hopes on these so-called *software agents*, which "know" users' interests and can act autonomously on their behalf. Instead of exercising complete control, people will be engaged in a cooperative process in which both human and computer agents initiate communications, monitor events and perform tasks to meet users' goals. In essence, we need software agents because :-

---

[3] The absence of proactivity is passiveness.

- more and more everyday tasks are computer-based
- the world is in a midst of an information revolution, resulting in vast amounts of dynamic and unstructured information
- increasingly more users are untrained
- and therefore users require agents to assist them in order to understand the technically complex world we are in the process of creating.

The number and type of application domains in which agent technologies are being applied to or investigated include workflow management, network management, air-traffic control, business process re-engineering, data mining, information retrieval/management, electronic commerce, education, personal digital assistants (PDAs), scheduling/diary management, etc. Indeed as Guilfoyle (1995) notes,

"in 10 years time most new IT development will be affected, and many consumer products will contain embedded agent-based systems".

The Intelligent Agents Group has focused on three research areas, which we believe span the key agent application domains of interest to our sponsors, namely, Intelligent User Interfaces, Distributed Agent Technology and finally, Mobile Agent Technology. These areas were also chosen in line with the three main agent attributes stated above, namely, learning, cooperation and mobility. The following three sub-sections will briefly summarise the main aspects of these areas in turn.

2.3 Intelligent User Interfaces

Before considering this field in any detail it is worth setting the scene by attempting to clarify exactly what is meant by the term "Intelligent Interface Agent". The first point to note is that there is a pre-existing body of work concerning Intelligent User Interfaces (IUI). It doesn't seem unreasonable to characterise such systems as being user adaptive. By adaptive it is meant of course that such systems attempt to modify their behaviour to maximise the productivity of the current user's interaction with the system. Obviously, for any such adaptation to be viable, it must be based upon information gathered concerning the current users behaviour. Given this, three interesting questions arise.

- What aspects of the user's behaviour are useful to capture?
- What information such behaviour gives us on the actual intentions and preferences of the user?
- Finally, what use can be made of any such information captured?

Intelligent (User) Interface Agents are fairly recent developments that use an agent oriented approach to the construction of such systems. The major factors that distinguish Interface Agents from any other IUI is the fact that agents are *proactive* and enjoy a degree of *autonomy*. These properties could manifest themselves in a number of different ways, for instance, there are a number of systems which perform an information filtering role, some of which filter in an autonomous fashion, only presenting the user with information it considers to be of interest to them. Similarly, this same type of system can also be proactive in that it actively searches for information that it judges would be of interest to its users.

For a more detailed review on Intelligent User Interfaces, please read chapter 3.

2.4 Distributed Agent Technology

Distributed Artificial Intelligence is a sub-field of Artificial Intelligence which is concerned with a society of problem solvers or agents interacting in order to solve a common problem: computers and persons, sensors, aircraft, robots, etc. Such a society is termed a ***Multi-Agent System***, namely, a network of problem solvers that work together to solve problems that are beyond their individual capabilities. The increasing interest in MAS research is due to the significant advantages inherent in such systems, including their ability to (a) solve problems that may be too large for a centralised single agent, (b) provide enhanced speed and reliability and (c) tolerate uncertain data and knowledge.

The problem solving performed by agents in a MAS is termed ***Distributed Problem Solving*** and involves research in the areas of coordination, negotiation and communication. In order for a MAS to solve common problems coherently, the agents must ***communicate*** amongst themselves, ***coordinate***

[4] The absence of reactivity is deliberativity.

their activities and ***negotiate*** once they find themselves in conflict. Conflicts can result from simple limited resource contention to more complex issue-based computations where the agents disagree because of discrepancies between their domains of expertise. Coordination is required to determine organisational structure amongst a group of agents and for task and resource allocation, while negotiation is required for the detection and resolution of conflicts.

For a more detailed review on Distributed Agent Technology, please read chapter 4.

2.5 Mobile Agents

Based on existing literature [Chess et al, 1995a; White, 1995; Nwana, 1996; URL1] we believe that the following definitions sufficiently characterise the essence of a mobile agent system.

A mobile agent is a software entity which exists in a software environment. It inherits some of the characteristics of an Agent (as defined in section 2.2). A mobile agent must contain all of the following models: an agent model, a life-cycle model, a computational model, a security model, a communication model and finally a navigation model.

This defines a mobile agent. However we must also consider the software environment in which mobile agents exist. We call this the mobile agent environment. The following is a definition of a mobile agent environment:

A mobile agent environment is a software system which is distributed over a network of heterogeneous computers. Its primary task is to provide an environment in which mobile agents can execute. The mobile agent environment implements the majority of the models which appear in the mobile agent definition. It may also provide: support services which relate to the mobile agent environment itself, support services pertaining to the environments on which the mobile agent environment is built, services to support access to other mobile agent systems, and finally support for openness when accessing non-agent-based software environments.

The above definitions state the essence of a mobile agent and the environment in which it exists. The mobile agent environment is built on top of a host system. Mobile agents travel between mobile agent environments. They can communicate with each other either locally or remotely. Finally communication can also takes place between a mobile agent and a host service.

For a more detailed review on Mobile Agent Technology, please read chapter 5.

2.6 Summary

The purpose of this chapter was to introduce the notion of what an agent is and the reasons for its increasing popularity. We also outlined the key aspects of the Intelligent Agents Group three research areas, each of which will be discussed in more detail throughout the remainder of this document.

3. Adaptive user interfaces

3.1 Introduction

This chapter gives an overview of existing work in the field of "Intelligent Interface Agents". This chapter begins with this introduction. Section 3.2 discusses some issues connected with the building of user models. Section 3.3 critically evaluates existing learning techniques with relation to the construction of user models. There then follows a section which presents a structured overview of existing User Interface Agents. Section 3.4 concludes this chapter by describing possible future applications for Intelligent Interface Agents.

The remainder of this introduction sets the scene for the rest of the review by attempting to clarify exactly what is meant by the term "Intelligent Interface Agent". The first point to note is that there is a pre-existing body of work concerning Intelligent User Interfaces (IUI). It doesn't seem unreasonable to characterise such systems as being user adaptive. By adaptive it is meant of course that such systems attempt to modify their behaviour to maximise the productivity of the current users interaction with the system. Obviously, for any such adaptation to be viable, it must be based upon information gathered concerning the current users behaviour. Given this, three interesting questions arise.

- Which aspects of the user's behaviour is it useful to capture?.
- What information does such behaviour gives us on the actual intentions and preferences of the user ?
- Finally, what use can be made of any such information captured?

Intelligent (User)Interface Agents are fairly recent developments that use an agent oriented approach to the construction of such systems. The major factors that distinguish Interface Agents from any other IUI is the fact that agents are *proactive* and enjoy a degree of *autonomy*. These properties could manifest themselves in a number of different ways, for instance one set of agents discussed below undertakes an information filtering role based upon perceived user interests. This role may involve actively seeking information with the filtering is undertaken with limited or no intervention on the part of user. In addition to these general properties it is usual for an interface agent to fulfil at least some of the following roles :-.

- *Assisting the user in communicating their task to the rest of the system*. This typically involves presenting the user with an easy to use interface which hides from them the actual underlying system which may be very complex. This should also provide benefits to system developers, allowing them to easily increase the functionality of the existing system by simply slotting in another functional component.

- *Learning the user profile*. The system interacts with the user via the interface that the interface agent provides. Thus this is the logical component of the system to attempt to build up a profile of the user. This should be based upon the user's behaviour in terms of interactions with the interface and the agent's knowledge of the semantics attached to the individual interface components.

- *Selecting for presentation components of the system's functionality*. This should be consistent with the user profile if available and users current interaction with the user interface. The agent is also responsible for presenting them to the user in a timely appropriate and accessible manner.

Another somewhat controversial aspect of agent based IUIs is the whole issue of *personification*. The persona of an agent is the visible presence of the agent from the users perspective. At one extreme several people are working on highly anthropomorphised agents which attempt to convey the whole range of human emotions [Bates 1992]. At the other extreme a number of Human Computer Interaction(HCI) workers lead most prominently by Schneiderman [Scheiderman 1997] are somewhat opposed to agent based interface solutions and particularly to the personified type, claiming they remove user control and are distracting. Of course a large number of people fall between these two extremes, one prominent exponent of the agent based approach who doesn't try to highly anthropomorphise agents is Maes [Maes 1997].

Of course there exists a myriad of different possible technologies and architectures which could be used to implement any kind of agent based interface system, some of which are described in the rest of this document.

3.2 How To Build a User Model

One of the principal problems in constructing an Intelligent Interface Agent is gathering accurate information regarding the user's interests, goals and general preferences. Sections 3.2.1 & 3.2.2. attempt to give a brief overview of existing Machine Learning techniques which might be useful in achieving this task.

3.2.1 Traditional Machine Learning Techniques

3.2.1.1 Symbolic Classifiers

Symbolic Classification attempts to classify a set of examples into one of a finite number of abstract classes. For instance in the example dataset shown below in Table 3.1, we have three examples each in a separate class. A typical example of a symbolic classifier system is Quinlan's C4.5 [Quinlan, 1987]. This system, a development from his earlier work on ID3, attempts to build a decision tree based upon a set of training examples each element of which consists of a number of possibly relevant attributes and the class that the example should be classified into.

Table 3.1 *An Example Dataset*

No of Legs	*Backbone*	*Hair*	*Wings*	*Class*
4	No	No	Yes	Insect
2	Yes	Yes	No	Mammal
4	Yes	No	No	Reptile

Given this training set of this type the algorithm produces a decision tree by selecting at each level of the tree an attribute on which to split the remaining set based on a measure of the information content of that attribute (termed the entropy), this heuristic measure is derived using Shannon's Information Theory.

3.2.1.2 Sub-Symbolic Classifiers

The sub-symbolic approach is based upon Neural Network systems. These systems take patterns as their input and classify them into one of a finite number of categories. These systems can be further split into two sub-categories based upon the architecture of the network and the method used to train them.

The first category shares many of the properties of the symbolic classifier systems such as that described above. These are termed "Supervised Neural Networks" and are typically trained using a set of input patterns for which the desired output category is known in advance. Some measure of the error between the actual and desired output is calculated which the training algorithm attempts to minimise as the training set is repeatedly presented. The classic example of a Neural Network architecture is the FeedForward Network trained using an algorithm such as Back Propagation [Rumelhart et al 1986]

The second class of Neural Network system requires no teacher to tell them the correct classification for the training examples and hence are termed unsupervised Neural Networks. Examples of this type of network are the ART (Adaptive Resonance Theory) Networks [Carpenter & Grossberg, 1995]. These networks fall into the general class of competitive learning techniques. Typically consisting of 2 layers, the first layer accepts input patterns, normalises this input and feeds it forward to a second layer. A node in the second layer is selected based upon the maximal activation and is put forward as the current hypothesis. Following this a matching phase occurs, where the activated node representing the hypothesis is matched with input I, the quality of the match is assessed and if a certain threshold is not passed, Layer 2 is reset and the input I activates a new node and the process is repeated. Thus the system can be said to develop its own classifications. Another network architecture of this type is the Self Organising Feature Map [Ritter 1995] which also learn to cluster data in an unsupervised fashion.

3.2.2 Learning Directly From the Environment

All the traditional techniques described above share one common feature, with the possible exception of the ART networks namely the learning is done off-line. The techniques described in this section are founded on the premise that the learning will be done in real time upon a situated autonomous agent. This adds additional complexities to the learning task, most significantly the information needed to learn a particular concept or task must be available in the agents perceptual world.

Reinforcement Learning

Another class of unsupervised learning algorithms is the reinforcement class of learning algorithms. One fairly recent example of this type of approach to learning is Watkin's Q Learning algorithm [Watkins 1989]. Q Learning works by calculating an estimate for state-action pairs *Q(s,a)*, which are defined to be the expected discounted sum of taking action a in state s and pursuing an optimal policy from there on in. Once these values are learnt the correct course of action can be determined at any state by taking the action with the highest *Q(s,a)* value. These *Q* values are estimated on the basis of experience according to the *Q* Learning algorithm. This algorithm is guaranteed to converge to the correct *Q* values if the environment is static and depends on the current state and action taken in it i.e. for Markovian worlds. In practice although any action could be taken in a given state, an action selection strategy called the Boltzmann distribution strategy is often used which ensures sufficient exploration whilst still favouring actions with higher value estimates.

In addition to the limits mentioned above, this mechanism also tends to be prohibitively slow for anything but the most simple real world problems. Thus a number of people have suggested improvements to the basic Q learning algorithm.

Learning by Observation

Also known as imitative learning, this learning technique is based upon the type of learning undertaken by many animats. As noted above, in theory we could learn how to perform any task simply by applying reinforcement learning given sufficient time. However if there is an available 'expert' who already knows how to complete the task or even some of the sub-tasks involved, we could possibly make use of this experience and speed up the learning process by imitating the expert and thereby gaining knowledge of how to tackle the tasks it is performing.

It should be noted that this approach does not assume an active teacher, the student is expected to learn about the task simply by watching and copying experts as they display their usual behaviour. The interested reader will find more information on this class of approach in [Mazur 1994]

Instructional Learning

This term encompasses a variety of techniques. At one extreme simple techniques which involve no explicit communication such as an expert attempting to demonstrate to a student how to perform a particular task without explicit communication could be classed as instructional techniques. Whilst at the other extreme quite complex communication languages might be employed to exchange information between agents see [Werner 89] for more information.

Other Possible Techniques of Interest

One area of possible interest is social psychology and the work done on attribution theory. In particular the work of Kelley et al on Causal schema, templates which people seem to use for making goal attributions based upon limited information seems potentially relevant.
Another area of work with possible relevance to the learning of user preferences is the whole area of Case Based Reasoning(CBR). In a multi-user system the period of interaction with an individual user will probably be fairly short. This is a potential problem as learning techniques tend to demand a fairly sustained period of interaction. One possible way of tackling this problem is instead of considering individual users we consider *classes of user* as our fundamental units. This would require an initial classification of users into one of a number of stereotypical user groups which could then be specialised to suit the individual user based upon whatever information could be gleaned during his/her limited interaction with the system. More interestingly, each user group could be treated as a kind of virtual user with individual users interactions forming parts of a much longer session of interaction. This way we could achieve the kind of length of interaction necessary to adapt the group stereotypes. This kind of scheme appears to be analogous to the kind of scheme typically employed by CBR systems. In these systems, there is a set of generalised cases which are specialised to fit the individual problems. If these specified cases differ significantly cases from the others in the case library, they too can be generalised and added to the library.

3.2.3 Evaluation Of Techniques

Before considering what learning techniques might be suitable for an Intelligent Interface Agent, it is worth considering what types of information it is desirable to learn. Such information might include at least the following :-

- User Interests
- Users Plans and Short Term Intentions
- Domain Knowledge i.e. knowledge about the system being interfaced
- User Preferences
- User Ability.

Each of these types of information raises particular problems and it seems most unlikely that a single learning technique will suffice for all. It is possible that the acquisition of user interests for instance can be achieved in a more simplistic manner than a more complex learning problem such as inferring a users goals and/or plans to achieve such goals.
Of the available techniques, it is felt that the traditional classifier techniques are unsuitable in this domain which is basically unsupervised in nature. The unsupervised neural networks might have a role in classifying users if some sort of stereotypical system is employed. This being the case, we are left with a problem as the other techniques discussed above are either new and untested or very slow and thus probably unsuitable for a real time system such as an intelligent user interface.
It would seem at first glance that the basic concept of stereotypical user groups and the associated techniques of Case Based Reasoning is one possible avenue for fruitful research and this is discussed in more detail later in this chapter. First however we present some examples of existing systems to give a flavour of what is possible with Intelligent Interface Agents.

3.3 *Existing Intelligent Interface Agents*

3.3.1 Some Broad Agent Classifications

Before moving on to consider some examples of existing work in this field we would briefly like to consider some general agent classifications as they apply to Adaptive Interfaces. Firstly when considering agent based systems it is often worth considering whether they are multi-agent or single agent in nature. Agents excel when there are multiple agents each with its own tasks and capabilities operating on distributed applications. The case for single agents operating on a single knowledge source is less clear cut, indeed it is debatable whether such an entity is an agent at all.

Secondly when considering the example systems that follow, it is important to bear in mind the characteristics which we have selected to characterise agent systems namely autonomy, adaptivity and co-operation. It is not enough simply to be autonomous as almost any software system can claim this. A successful agent system should ideally show aspects of all three of these characteristics and as a minimum two out of the three.

3.3.2 Area 1 Information Filtering Agents

Figure 3.1 below, outlines a generalised reference architecture for Information Filtering Agents. The remainder of this sub-section details some example systems which I feel fall into this category of system.

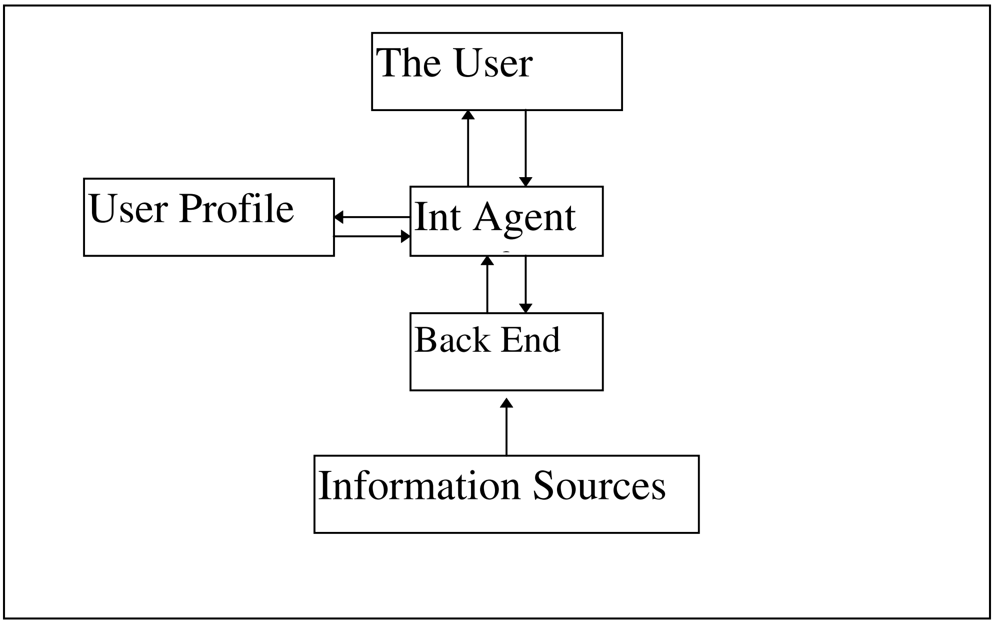


Figure 3.1 *Reference Architecture for Information Filtering Agents*

It may be noted that the interface agent in this reference diagram is responsible for providing the interface to the overall system and is separate from the content provider. This is the ideal position, unfortunately however a lot of systems in this class seem to combine the system functionality with the actual interface agent. Our preferred model for systems with an interface agent component is based upon the assistant metaphor. Where the agent is separate from the system and can act as a conduit between system and user, assisting in the users interaction with the system.

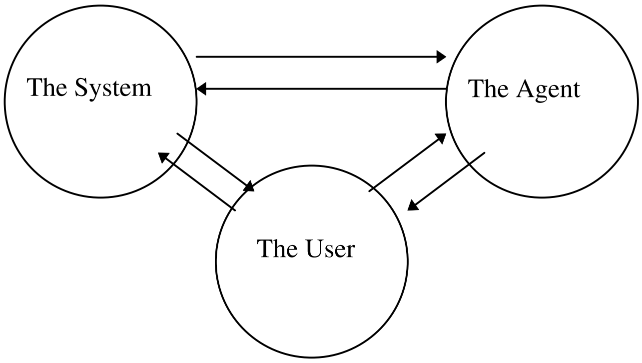


Figure 3.2 General Interface Agent Schemata

The Filtering agent is probably one of the simplest classes of Intelligent Interface Agents imaginable, indeed some would question if this class of system exhibits intelligence at all. This class of system typically consists of a number of large information sources that are to be filtered in some way to leave only the information relevant to the user.

Two major approach to the construction of such systems exist, those which utilise explicit user models and those which do not. Of those systems which do not utilise explicit user models the most effective technique by far is Automated Collaborative Filtering (ACF).This is used in a number of systems including HOMR, an MIT system for recommending musical CD's. Collaborative Filtering systems also known as Recommender Systems, make use of other users of the system classifying items as being of interest or not and then identifying the current user with similar past users and using their expressed preferences to select items of interest for the current user.

Although clearly useful in some situations, even the authors of Homr admit that for larger, less uniform domains than musical taste some kind of feature based system is required to partition the domain before applying ACF. This approach is adopted in their Webhound [URL13] System.

There are three broad classes of user model that could be used to categorise the types of user model currently in use, based upon the duration and type of interaction with the user:

- An explicit *user profile* based upon a series of questions designed to accurately acquire from the user their interests and preferences.
- Based upon relatively short term interaction with a system which monitors user behaviour a *shallow model* of user preferences and interaction patterns.
- Based upon long term interaction with a system which monitors user behaviour, a *deep model* which captures users interests and goals.

The third type of model is probably the most desirable in this context as it does not distract the users from their main task of acquiring the information they require. However it is notoriously difficult and unreliable to derive information in this manner. Explicit gathering of information also suffers from problems however. It is intrusive, yet there is no guarantee that the questions asked will be answered truthfully, or even that the questions asked are the right ones for obtaining the information desired. Shallow preference models probably are not of much use in this type of system which is concerned with deeper modelling of interests. There is now a fairly large number of agents that fall into this class of system, email and news group filtering being very popular application domains. One interesting example of this type of system is the work of Jennings and Higuchi [Jennings & Higuchi 1993]. Their system uses an Artificial Neural Network (ANN) to classify news group articles into interesting and irrelevant ones and is trained by users reading or rejecting articles suggested by the system which is then given as feedback to the neural network. Systems of this sort suffer from a number of disadvantages. First, the systems almost always base their understanding of the content of the news articles or other information sources on features or keywords. These are known to be unreliable in capturing the true meaning of natural language texts. Second, even if the features selected can effectively represent the pieces of information to be filtered there is no guarantee that we can successfully deduce the users' interests based upon these features. Another interesting system of this type also from MIT is Maes Collaborative Interface Agent system [Lashkari et al, 1994] which attempts to facilitate learning between different User Interface Agents which are serving similar users by allowing such agents to communicate with one another. Other systems that attempt to intelligently filter information include Stanford's SIFT system. [Yan and Garcia-Molina 1994] .

3.3.3 Information Retrieval Agents

Although the architecture diagram shown in Figure 3.3 may appear at first glance to be rather similar to that of the first class of system discussed in the previous section, the inclusion of the arrow indicating a flow of communication from the interface agent to the information resources reflects a significant step

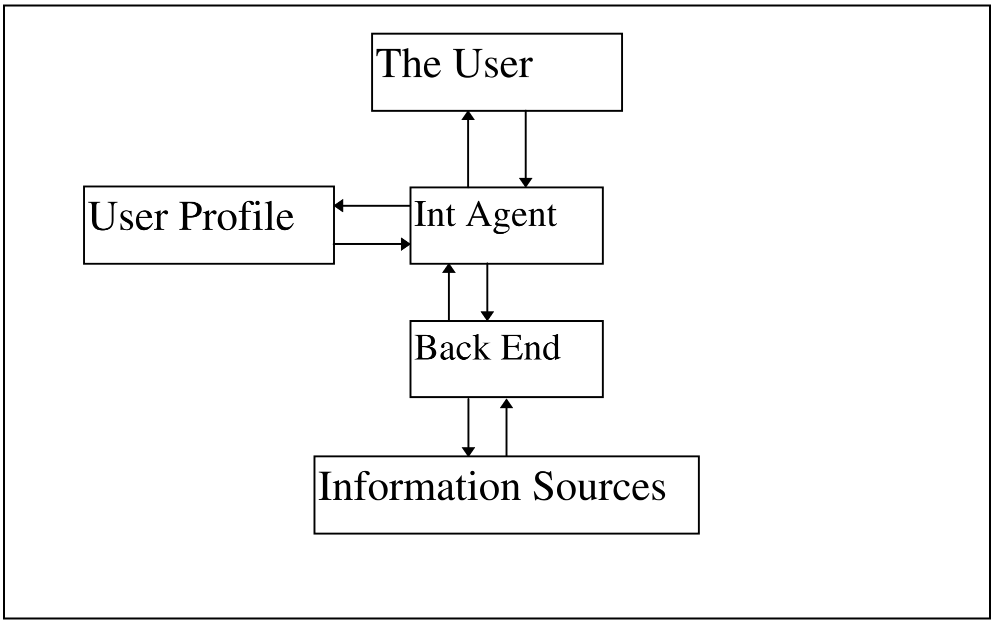


Figure 3.3 A Reference Architecture for Information Retrieval Ag*ents*

up in sophistication. This is the first class of system that exhibits a significant degree of proactivity. The extra arrow on the reference architecture indicates that the interface agent requests particular pieces of information from the information sources, rather than passively filtering information presented to it. This approach lends itself particularly nicely to being implemented using a multi-agent system. Using this approach each separate information source would be modelled by an agent and the interface agent would broadcast a request for information. Upon receipt of such a request each information agent would reply with a message containing all the information they have access too with relevance to the initial request. The interface agent, having received all these separate pieces of information, is responsible for aggregating the information sensibly and presenting the information in a unified and easy to access fashion. This approach brings with it a number of benefits to compensate for the added complexity. Chiefly it allows us to deal with distributed sources of information effectively, for example the World Wide Web. It also brings benefits in terms of scalability in that the problems associated with incorporating a new knowledge source are greatly reduced as we simply need to design a new agent and slot it in. Systems of this complexity are very thin on the ground. It is clearly of importance that the information requested satisfies the users query or goals. Thus the request sent by the interface agent must be based both on the information obtained directly from the interface and also upon any user model the system has managed to build up. One system that aspires to adopt this kind of approach is the KIMSAC project [Charlton 1997]. This involves the provision of multimedia kiosks at several sites around Ireland drawing social welfare and employment information from a number of distributed knowledge sources. Other systems such as BargainFinder[URL11] also have the potential to be developed as instances of this class of system, although they lack at the moment any attempt at User

Modelling. Finally SRI have created a tourist application drawing information from various sources to provide a map based tourist information service for the San Francisco area. [Moran et al 1997].
A number of systems attempt to provide virtual newspapers which draw information from a number of different sources based upon a user's expressed interests. This type of system differs from the traditional newspapers in that the user of the system explicitly chooses what information should be present in their own personal newspaper. Thus the newspaper is sensitive to each user's unique information needs but the creators of the system loose overall editorial control. Typically content information is retrieved from real-time news wires as well as from other less dynamic sources. This allows the newspapers to remain up to date although typically at the expense of the in-depth analysis of the relevant issues mentioned earlier. An example of this type of system is Crayon[URL12] which presents news from a number of different user selected categories the presented information being drawn from numerous sources. Netscape also provide access to a number of similar services via its Inbox facility as do Yahoo. Another similar service is provided by Excite. Newshound provides a somewhat different service which is charged for. Here the users enter keywords which tell the system what news items they are, possibly are and definitely are not interested in. Again the system returns news articles drawn from a variety of online news sources (mainly newswires). Although all of the above systems could be viewed as agents they are rather simplistic making use of explicitly entered keywords or features. More interesting problems would be to try and automatically derive such interests from the users actions or to try and adapt individual news items to better suit the readers own viewpoint. However the latter of these tasks would be by no means trivial and perhaps is beyond current technology.

3.3.4 Personal Digital Assistants (PDAs)

This class of system shown in Figure 3.4 embodies what is probably the dominant type of interface agent, ie. the intelligent assistant. That is a system which attempts to help users perform their routine

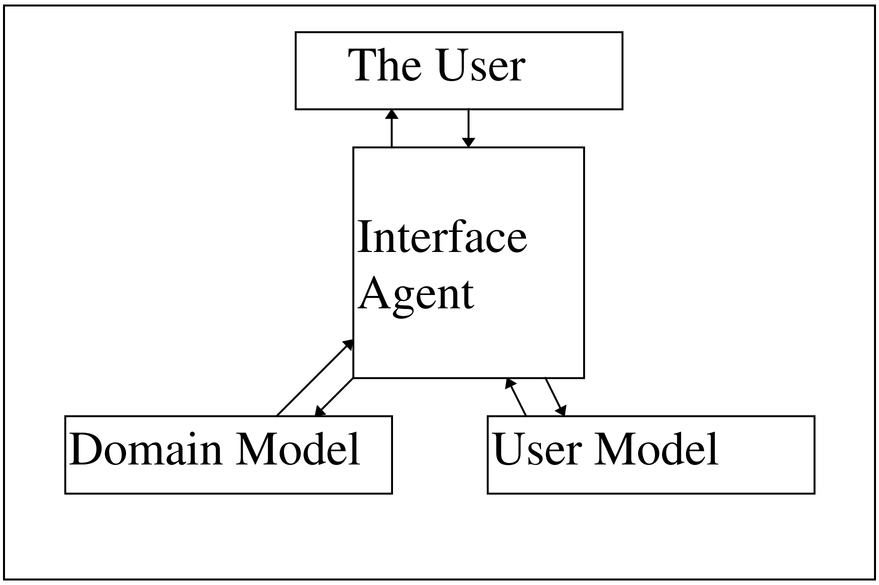


Figure 3.4. Reference Architecture for PDA Agent.

day-to-day tasks. This type of system would ideally have domain knowledge ,i.e. knowledge of the capabilities of the system it is assisting the user to use, as well as a model of the users abilities and preferences. The agent is then responsible for providing help and/or advice based upon this knowledge as well as the user's current task as perceived by the agent. For instance Lester [Lester et al 1997] in their "Design a Plant" system (a Pedagogical system designed to allow children to learn about plant biology by designing their own plants to fit particular environments) include a highly visible agent in the form of a nerdy alien which gives advice to the students when the system deems they need help. Other interface agents attempt to automate the user's repetitive tasks for instance Etzioni [Etzioni & Weld 1994] has done some work in the UNIX operating system domain.

More closely related to the Internet, Fisher and Thomas [Fischer & Thomas 1997] have produced a system to help a user organise their bookmarks effectively, whilst Pandit and Kalbag [Pandit & Kalbag 1997] at Intel have produced an agent based system which provides the user with links based upon keywords which contain email addresses, URLs and geographic place names amongst other things.
These last two systems raise some interesting issues regarding research into Intelligent Interface systems. Whilst both systems are clearly useful systems which would undoubtedly increase the user's efficiency at performing their day to day work, the agents involved perform relatively simple tasks which require little if any intelligence to achieve and certainly don't require any adaptiveness. Is it realistic then to call such systems Intelligent Interface Agents ? Perhaps it would be better just to refer to them as Interface Agents, the use of the word "intelligent" creates expectations in the minds of users which current systems are probably incapable of fulfilling. This is not limited to agents but reflects a wider trend in the IUI community towards the use of simple AI techniques only. This type of interface agent is also the one that is of most relevance to the traditional telecommunications domain. The PANG project [Montelius et al 1996] attempts to provide call screening based upon a "Business Card" concept which are processed by an personal assistant agent which maintains a user profile. [Esfandiari et al] in their system for assisting in the network supervision task, also make use of interface agents which assist by automating some tasks learnt by watching over the shoulder of a human user. In general there would seem to be a role for some kind of interface agent to assist the user in dealing with complex possibly distributed systems

3.4 Conclusions

In conclusion, it would seem there is a role for Intelligent Interface Agents in the domain of Telecommunication Service Provision if this is taken in its broadest sense allowing the inclusion of the World Wide Web. Two main application domains stand out amongst the available literature. First, many systems exist which attempt an information filtering/retrieval role. These systems would appear to be quite successful in what they do, particularly the ones based on ACF clustering techniques. The ones based upon user models suffer from the fact that there doesn't exist a reliable way to extract meaning from arbitrary natural language text and therefore instead utilise keyword extraction.
There also exists a role for the assistant style of agent for complex systems particularly those based upon multi-agent systems. This could include either the information provision style of system typified by the KIMSAC project or a system which is more procedurally oriented such as the PANG project. Whatever the application, some degree of adaptivity is desirable in order to maximise the improvement in user productivity that the assistant is able to generate. Unfortunately most current agent based systems make little use of significant user modelling relying for the most part on quite simplistic user profiles. It would seem therefore that there is a need for more research into the use of user modelling in intelligent interface agents. Our work will concentrate at least initially on the information provision/filtering domain and will be centred around the case-based approach mentioned briefly above.

4. Distributed Agent Technology : An Overview

4.1 Introduction

Distributed Artificial Intelligence (DAI) is concerned with problem solving in which groups of agents solve tasks. In this chapter, we explore existing problem solving strategies and the applications to which they have been applied. Section 4.2 will first explain the concepts of DAI, Multi-Agent Systems and Distributed Problem Solving (DPS). Sections 4.3, 4.4 and 4.5 will then outline the various research topics associated with DPS in Multi-Agent Systems, namely, coordination, negotiation and communication respectively. Finally, section 4.6 will conclude this chapter.

4.2 DAI and Multi-Agent Systems

Research in Artificial Intelligence has aimed at developing software to simulate so-called intelligent capabilities of human beings such as reasoning, natural language communication, and learning: with such programs, the computer's role departs from that of a mere tool to progressively become a kind of assistant to humans. The implementation of computer networks was a big step toward the development of computer organisations or "societies", since collaboration between individuals requires that communication links be established and used effectively**.** ***Distributed Artificial Intelligence*** is a subfield of Artificial Intelligence which has, for more than a decade now been investigating knowledge models, as well as communication and reasoning techniques that computational agents might need to participate in "societies" composed of computers and people. More generally, DAI is concerned with a society of problem solvers or agents interacting in order to solve a common problem: computers and persons, sensors, aircraft, robots, etc. Such a society is termed a ***Multi-Agent System***, which can be defined as "a loosely-coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities" (O'Hare & Jennings, 1996).

The motivations for the increasing interest in MAS research includes their ability:-

- to solve problems that are too large for a centralised single agent to do due to resource limitations or the sheer risk of having one centralised system;
- to allow for the interconnecting and interoperation of multiple existing legacy systems, e.g., expert systems, decision support systems, etc.;
- to provide solutions to inherently distributed problems, e.g., air traffic control (Cammarata et al, 1993);
- to provide solutions which draw from distributed information sources;
- to provide solutions where the expertise is distributed, e.g., in health care provisioning;
- to enhance speed (if communication is kept minimal), reliability (capability to recover from the failure of individual components, with graceful degradation in performance), extensibility (capability to alter the number of processors applied to a problem), the ability to tolerate uncertain data and knowledge
- to offer conceptual clarity and simplicity of design.

Researchers usually distinguish between two types of Multi-Agent Systems [Bond and Gasser, 1988; Rosenschein & Zlotkin, 1994], depending on the degree of cooperation exhibited by the individual agents:-

- ***Cooperative Distributed Problem Solving or Cooperative Multi-Agent Systems (CMAS)***

  Early work in DAI was exclusively on interacting agents that had all been designed by a *single designer*. Therefore, agents could be *counted on to act for the greater good of the system*, since they could all be programmed that way by the designer, who was only concerned with increasing the general system's performance and not the performance of individual agents. Hence such agents are considered cooperative.

- ***Self-Interested Multi-Agent Systems (SMAS)***

  In the mid 1980s, a distinct research direction within DAI began to take shape. Certain researchers began to ask questions related to *individually motivated agents*, who had been designed by *independent designers.* When considering system behaviour, they could not count on agents cooperating just because they would be designed that way. What was important to each independent designer was the *benefit they could derive from their individual agents.* Hence such agents are considered self-interested, competitive or non-cooperative and may exhibit antagonistic behaviour.

The cooperative-to-antagonistic spectrum in a MAS has been surveyed by Genesereth et al (1986) and Rosenschein and Genesereth (1985). They note that the situation of total cooperation, known as the *benevolent agent assumption*, is accepted by most DAI researchers, but that it is not always true in the real world, where agents may have *conflicting goals*. Such conflicting goals are reflected, for example, by a set of personal meeting-scheduling agents where each agent tries to schedule a meeting at the best time for its particular owner.

The problem solving performed by agents in a MAS is termed ***Distributed Problem Solving*** and involves research in the areas of coordination, negotiation and communication. In order for a MAS to solve common problems coherently, the agents must ***communicate*** amongst themselves, ***coordinate*** their activities and ***negotiate*** once they find themselves in conflict. Conflicts can result from simple limited resource contention to more complex issue-based computations where the agents disagree because of discrepancies between their domains of expertise. Coordination is required to determine organisational structure amongst a group of agents and for task and resource allocation, while negotiation is required for the detection and resolution of conflicts. The following sections describe coordination, negotiation and communication in more detail and introduce several approaches devised to achieve them (see sections 4.3, 4,4 and 4.5 respectively).

4.3 Coordination in Multi-Agent Systems

Coordination is central to a MAS, for without it, any benefits of interaction vanish and the group of agents quickly degenerates into a collection of individuals with a chaotic behaviour. Coordination has been studied by researchers in diverse disciplines in the social sciences, including organisation theory, political science, social psychology, anthropology, law and sociology. For example, organisation theorists have investigated the coordination of systems of human beings, from small groups to large formal organisations [Galbraith, 1977; Thompson et al, 1991]. Even biological systems appear to be coordinated though individual cells or 'agents' act independently and in a seemingly non-purposeful fashion. Human brains exhibit coordinated behaviour from apparently 'random' behaviours of very simple neurones. Essentially, coordination is a process in which agents engage in order to ensure a community of individual agents acting in a coherent and harmonious manner.

There are several reasons why multiple agents need to be coordinated:-

- First, prevent *chaos*. No agent possesses a global view of the entire agency to which it belongs, as this is simply not feasible in any community of reasonable complexity. The chairman of Ericsson, for example, cannot possibly be aware of the detailed activities of all its employees! Consequently, agents only have local views, goals and knowledge which may interfere with rather than support other agents' actions. Coordination is vital to prevent chaos during conflicts.
- Second, to meet *global constraints.* Agents performing network management may have to respond to certain failures within seconds and others within hours. Coordinating agents behaviours is therefore essential to meet such global constraints.
- Agents in a MAS possess different capabilities and expertise. Therefore, agents need to be coordinated in just the same way that different medical specialists, including anaesthetists, surgeons, ambulance personnel, nurses etc. need to coordinate their capabilities to treat a patient.
- Agent's actions are frequently interdependent and hence an agent may need to wait on another agent to complete its task before executing its own. Such interdependent activities need to be coordinated.

The following subsections outline the key approaches to solving the coordination problem in MASs.

4.3.1 Organisational Coordination

The easiest way of ensuring coherent behaviour and resolving conflicts seems to consist of providing the group with an agent which has a wider perspective of the system, thereby exploiting an organisational or hierarchical structure. This is the simplest coordination technique and yields a classic master/slave or client/server architecture for task and resource allocation among slave agents by some master agent. The master controller could gather information from the agents of the group, create plans, and assign tasks to individual agents in order to ensure global coherence. Some such systems employ a blackboard architecture to achieve coordination, such as Werkman's DFI system [Werkman, 1990] and the Sharp Multi-Agent Kernel (SMAK) system [Kearney et al, 1994]. In this scheme, the blackboard's knowledge sources are replaced by agents who post to and read from the general blackboard. The master agent schedules the agent's reads/writes to/from the blackboard. In the DVMT system, which also exploits a blackboard architecture [Lesser & Corkill, 1983], coordination also occurs amongst peers. Centralised markets, such as MAGMA [Tsvetovatyy & Gini, 1996], employ a global blackboard for posting buying and selling prices.

However, such an approach is impractical in realistic applications because it is very difficult to create such a central controller informed of all agents' intentions and beliefs. Durfee et al (1989) point out that such centralised control as in the master/slave technique is contrary to the basic assumptions of DAI.

4.3.2 Contracting for Coordination - The Contract Net Protocol

The most renowned coordination technique for task and resource allocation among agents and determining organisational structure is the *Contract-Net Protocol (CNP)* [Smith & Davis, 1981; Davis & Smith, 1983].

In this approach, a decentralised market structure is assumed and agents can take on two roles, a *manager and contractor*. The basic premise of this form of coordination is that, if an agent cannot solve an assigned problem using local resources/expertise, it will decompose the problem into subproblems and try to find other willing agents with the necessary resources/expertise to solve these subproblems. The problem of assigning the subproblems is solved by a *contracting mechanism* consisting of contract announcement by the manager agent, submission of bids by contracting agents in response to the announcement, and the evaluation of the submitted bids by the contractor, which leads to awarding a subproblem contract to the contractor(s) with the most appropriate bid(s) (see Figure 4.1). The CNP has been used in many applications, such as Parunak (1987), who applied it to manufacturing control.

Although CNP is considered by Smith and Davis as well as many DAI researchers to be a *negotiation* principle, Jennings (1996, pg. 212) and us believe it is more a standardised *coordination* method for the following reasons:-

- In the view of Smith and Davis, there must not be a conflict at all between the agents to start the CNP. Hence there is no possibility of *bargaining* between the agents. The manager does not communicate its minimal condition, nor do the bidders have a second choice (no constraint relaxation)
- A mutual decision is eventually given by the decision of the offer, hence there is no two-way agreement.

Other approaches [Conry et al, 1986] have elaborated on the contract net to build more effective negotiation by including constraint relaxation.

Some of its advantages include dynamic task allocation via self-bidding which leads to better arguments, agents can be introduced and removed dynamically, it provides natural load-balancing (as busy agents need not bid) [Huhns & Singh, 1994]. Its limitations involve the fact that it does not detect or resolve conflicts (as discussed above), the agents in the contract net are considered benevolent and non-antagonistic (which in real-world scenarios is not realistic), and finally it is rather communication-intensive, the costs of which may outweigh some of its advantages in real-world applications.

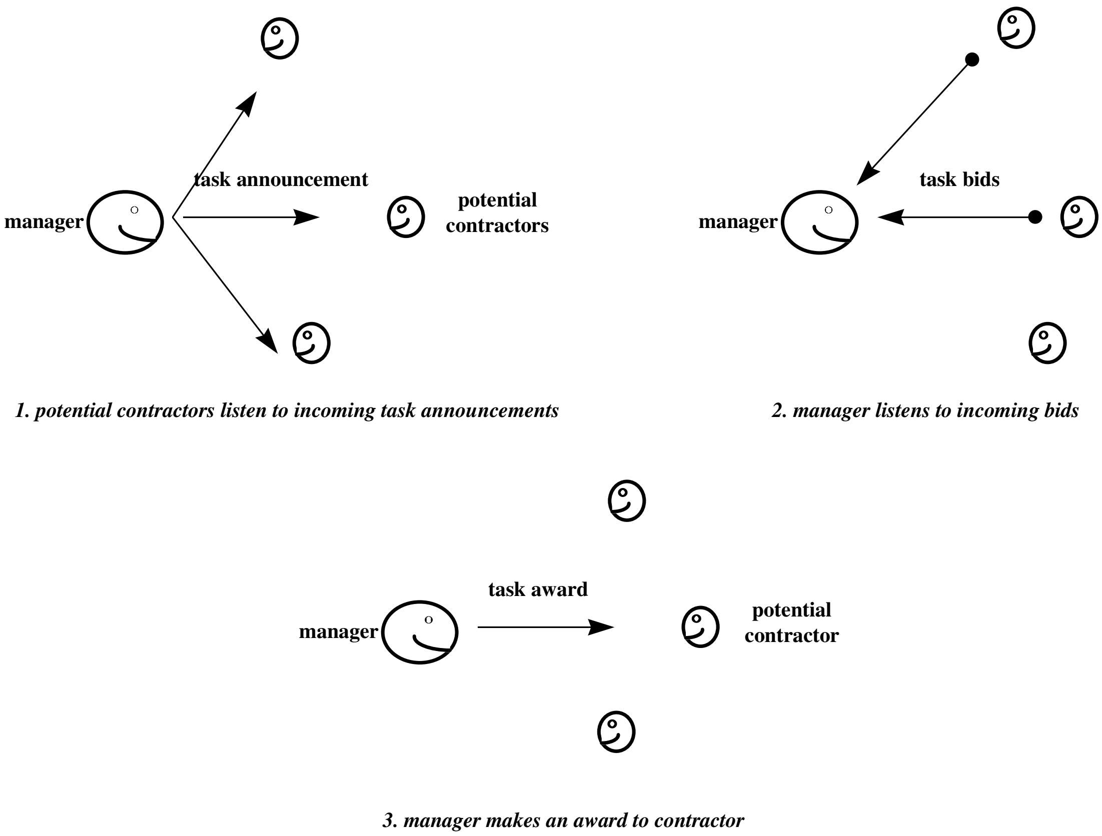


Figure 4.1 *The Contract Net Protocol*

4.3.3 Multi-Agent Planning for coordination

More traditional AI research has led to viewing the problem of coordinating multiple nodes as a planning problem. Multi-agent planning emphasises certain avoidance of inconsistent and conflicting situations, which is critical in applications such as *air-traffic control.* By forming a multi-agent plan, the nodes determine all of their actions and interactions beforehand, leaving nothing to chance. There are two types of multi-agent planning, namely, centralised and distributed:

In centralised multi-agent planning, the separate agents form their individual plans and then send these plans to a central coordinator, who analyses them and finds potential plan conflicts [Georgeff, 1983]. The idea behind this approach is that the central coordinator can (a) identify critical regions of plans around which agents should synchronise and (b) insert plan steps for sending and waiting for synchronisation messages to ensure proper synchronisation. The individual partial plans can then be merged into a multi-agent plans with conflicting interactions eliminated. Cammarata et al (1983) employ centralised multi-agent planning in a simulated air-traffic control domain. In this system, agents (i.e., the aeroplanes) in a potential conflict scenario, e.g., two aeroplanes are about to conflict, decide on one of their number to act as the coordinating agent to whom the plans of the other agent are sent. This coordinating agent will then attempt to modify its own flight plan in order to resolve the conflict. A disadvantage of this technique is that having the agents first form their plans as if they were acting alone and then coordinating these plans can miss opportunities for cooperation that would have been possible had the agents built their individual plans concurrently with reasoning about what other nodes are doing. It also shares many of the disadvantages inherent in the master/slave approach to coordination.

The distributed technique for multi-agent planning foregoes the use of a central coordinator, and instead allows agents to model each other's plans [Corkill, 1979; Georgeff, 1984; Konolige, 1983]. Agents communicate in order to build and update their individual plans and their models of others' until all conflicts are removed. An example of this approach is Lesser and Corkill's functionally accurate and cooperative (FA/C) protocol (Lesser & Corkill, 1981). In this approach, loosely coupled agents form high-level (but possibly incomplete) plans, results and hypotheses which they exchange with each other. Next, they refine these until they all converge on the same global complete plan. This approach has been used in Lesser and Corkill's Distributed Vehicle Monitoring Testbed (DVMT) - a system for testing coordination strategies.

Both approaches to multi-agent planning require that nodes share and process substantial amounts of information and hence multi-agent planning generally involves more computation and communication than other approaches.

4.3.4 Social Laws for Coordination

Researchers attempting to coordinate intelligent agents in domains like air traffic control or urban traffic discovered that coordination is easier in *routine than in unfamiliar situations* [Chaib-draa, 1997].

If all agents had complete knowledge of the goals, actions and interactions of their members, it would be possible to know exactly what each agent is doing at the present moment and what it is intending to do in the future. In this context, it would be possible to avoid conflicting and redundant efforts, agents could be perfectly coordinated and the effort of achieving this state would be prohibitively high.

However, such complete knowledge about actual actions and reactions is only feasible in routine situations. For example, in an urban traffic context, a routine situation is where the car in front signals that he is turning right, or when the traffic lights go green, so one can coordinate one's actions, based on knowing what others are doing *according to social law (*traffic laws). In real-world domains, there are also familiar and unfamiliar situations. A familiar situation is not routine. A familiar situation might occur when the traffic lights are broken and a policeman is directing the traffic. Again, based on social law, his directions will coordinate the traffic well. An unfamiliar situation might be where the traffic lights are down and there is no policeman directing the traffic.

Coordination decreases as the situation becomes less familiar, as more analysis and reasoning is required, which is laborious and could result in conflicts between agents. Therefore, Chaib-draa argues that agents should be given social laws, so that their processing relies on skills, which results in fast, effortless and is propitious for coordinated activities between agents. Agent performance is governed by stored patterns of predefined procedures, that map directly from observation (i.e., perception) to an action. Unfamiliar situations are adapted to familiar situations, using case-based reasoning. The agent model consists of three levels, skill (routine), rules (familiar), knowledge (unfamiliar). The low levels, routine familiar situations are strengthened by enriching each agent with *social regularities*, such as, coordination rules (drive on left, speak in turn, etc.), cooperative rules (recycling cans or consuming energy: actions by one agent makes sense only if other agents also perform action) and social collections (e.g., roles, groups, organisations, etc.).

4.4 *Negotiation Techniques*

An ongoing objective in DPS research is to capitalise on the insights about how humans coordinate their activities through negotiation. The common idea in all DAI contributions to negotiation is that agents use negotiation for conflict resolution and hence coordination. Though negotiation is highly important for the modelling of multiagent systems, there is no clear and common definition of what negotiation is. Indeed there are probably as many definitions of negotiation as there are negotiation researchers. Perhaps a basic definition is that of Bussmann and Muller (1992):-

"….negotiation is the communication process of a group of agents in order to reach a mutually accepted agreement on some matter."

Agreement might be about price, military arrangements, about a meeting place or time, about joint action, or about a joint objective. The search process may involve the exchange of information, the relaxation of initial goals, mutual concessions, lies or threats. Hence the basic idea behind negotiation is reaching a *consensus*.

Other researchers place a stronger definition on negotiation, arguing that in order to negotiate effectively, agents must reason about beliefs, desires and intentions of other agents [Rao and Georgeff, 1995]. This has lead to the usage of all sorts of AI and mathematical techniques including logic, case-based reasoning, belief revisions, distributed truth maintenance, optimisation and game theory. Therefore, it is not difficult to imagine the huge and varied literature on this subject of negotiation.

Negotiation can be *competitive (SMAS) or cooperative (CMAS)* depending on the cooperative behaviour of the individual agents involved (recall section 4.2):-

1. ***Competitive Negotiation in a SMAS.*** Used in situations where "agents of disparate interests attempt to make a group choice over well-defined alternatives" [Rosenschein & Zlotkin, 1994]. (e.g., the interaction between computers controlling and sharing resources of distinct and possibly competitive agencies - a computer in France and a computer in Germany that each control part of a linked telecommunications network may find it beneficial (or even necessary) to come to agreements about how packets are routed). Therefore, competitive negotiation involves independent machines with independent goals that interact with each other. They are not *a priori* cooperative, share information or willing to back down for the greater good, namely they are competitive.
2. ***Cooperative Negotiation in a CMAS.*** Used in situations where agents have "a global goal/single task envisioned for the system" [Smith & Davis, 1981]. Distributed systems that have been *centrally designed* to pursue a single global goal. These agents are sometimes called 'collaborative' [Chu-Carroll & Carberry, 1995]. Example domains in which this collaborative principle holds include (1) a university course advisement domain, where an advisor and student collaborate on developing a plan to achieve the student's domain goal, (2) the air-traffic control domain, where air-traffic control systems collaborate to establish the best routes for planes to land, (3) the battleship domain, in which multiple strategists collaborate on developing the best attack/defence tactics, and so on.

Therefore, the following sub-sections will outline both competitive and cooperative negotiation strategies and their applications.

4.4.1 Competitive Negotiation Using Game Theory

There is now a growing body of work on negotiation which is based on game theory [Luce & Raiffa, 1957]. The key researchers in this area are perhaps Rosenschein and Zlotkin (1984), who use tools of game theory to achieve coordination amongst a set of rational and autonomous agents without an explicit coordination mechanism built into these agents *a priori*. In other words, they do not assume the 'benevolent agent assumption'.

The key concepts in this game theory approach to negotiation are the following:-

- *Utility Functions.*
  Utility can be defined as the difference between the worth of achieving a goal and the price paid in achieving it.
- *Space of Deals*
  A deal is an action an agent can take which has an attached utility.
- *Strategies and Negotiation protocols*
  The negotiation protocol defines the rules which govern the negotiation, including how and when it ends (e.g., by agreement or without a deal). This research has introduced the *Unified Negotiation Protocol*, through which agents converge on joint plans, called semi-cooperative deals, that increases all of their expected utilities, despite the fact that each has a chance of not achieving its goals.

The actual negotiation proceeds is as follows. Utility values for each outcome of some interaction for each agent are built into a pay-off matrix, which is common knowledge to both (typically) parties involved in the negotiation. The negotiation process involves an interactive process of offers and counter-offers in which each agent chooses a deal which maximises its expected utility value. There is an implicit assumption that each agent in the negotiation is an expected utility maximiser. At each step in the negotiation, an agent evaluates the other's offer in terms of its own negotiation strategy.

However, as Nwana (1996) critiques:-

- agents are presumed to be fully rational and acting as utility maximisers using predefined strategies;
- all agents have knowledge of the pay-off matrix and therefore full knowledge of the other agent's preferences - this is certainly unlike the real world where agents only have partial or incomplete

knowledge of their own domains, let alone those of others. Therefore, this is unrealistic for truly non-benevolent and loosely coupled agencies.

- much of the work presumes two agents negotiating, whereas in reality quite a large number of agents will be typically involved in negotiation.

Therefore, despite Zlotkin and Rosenschein's provision of mathematical proof of their ideas, it is unlikely that theory-based negotiation will suffice for real-life, industrial agent-based applications.

4.4.2 Competitive Negotiation in Electronic Commerce

A rapidly growing segment of the Internet is Electronic Commerce. For sheer convenience and preservation of time, consumers are looking for suppliers selling products and services on the internet (e.g., physical and electronic goods; stocks and other investments). Meanwhile, suppliers are looking for buyers to increase their market share. The vast amount of information on the Internet causes a great deal of problems for both the consumer and the seller. It is speculated that Intelligent Agents will facilitate the solving of these problems.

The two systems below, namely, KASBAH and MAGMA, involve Personal Assistants (PA) representing buyers and sellers in a virtual marketplace and performing negotiation. The essential idea in both these systems is that users tell their PAs what they would like to be done ("sell this for the best possible price") and trust it to figure out how to accomplish this task, freeing their time and energy for more interesting pursuits. In addition, users hope that agents might be able to sell (and buy) goods better (e.g., at a higher price) than they would be able to, by taking advantage of their edge in processing speed and communication bandwidth. We will now briefly introduce the negotiation techniques employed in these two key electronic commerce applications:-

- KASBAH [Chavez and Maes, 1996]

  Kasbah is a classified ads service on the WWW that incorporates interface agents. Kasbah is meant to represent a 'market-place' (a web site) where Kasbah agents, acting on behalf of their owners, can filter through the ads and find those that their users might be interested in. The agents then proceed to negotiate to buy and sell items.

  The negotiation scheme for buyers and sellers involves "price-raise" and decay functions respectively. For sellers, the PA will offer the item at the highest desired price, and then decrease this price according to the decay function (which is user-specified as being linear, quadratic or cubic). So, when the desired date to sell the item arrives, the asking price should be about the lowest acceptable price. The converse is true with buyers and their "price-raise" functions. Users has control over its PA's actions: (a) they can check who their PAs have talked to and the prices involved, and hence can fine-tune their agents accordingly, (b) they have the "final say" before agents shake hands on the agreed deal. These user specifications for negotiation are carried out using HTML.

- MAGMA [Tsvetovatyy & Gini, 1996]

  MAGMA moves the marketplace metaphor to an open marketplace involving agents buying/selling physical goods, investments and forming competitive/cooperative alliances. These agents access a global blackboard, called an "offer board", that contains offers from other buyers and sellers. The negotiation scheme, like KASBAH above, involves sellers specifying to their PA, the highest price the item (physical good, investment or company) should sell at and posts this data to the offer board. The buyers indicate to their PA the category of item it is interested in and the number of best offers to retrieve. The buyer PA displays these offers and the buyer will select which offer to follow up. The buyer PA will then send an accept offer to the relevant seller agent.

Another area of interest in the Electronic Commerce domain is the concept of the *Virtual Enterprise*. Currently, the time interval within which a product can be marketed successfully is dramatically decreasing; therefore, enterprises are faced with hard terms of competition. Decreasing innovation cycles, changing market situations as well as growing specialisation in individual market segments demand new ways of economic thinking, increasingly forcing enterprises into cooperations or virtual enterprises, sometimes even with direct competitors. These cooperations enable enterprises to share skills, costs, access to one another's markets and resources and, at the same time, decrease the risk of investments. An example MAS, which attempts to enable individual resources via competitive negotiation to form virtual enterprises is described below:-

- AVE - Agents in Virtual Enterprises [Fischer et al, 1996]

  Negotiation occurs via an *auctioning mechanism* in AVE between individual competitors for a specific task. In this auction, the product/service manager has the role of auctioneer, the subjects of the auction are the individual partial processes forming the overall service, and the bidders are enterprises that are interested in contributing these partial processes to the virtual enterprise. There are several different auction mechanisms proposed [Murnighan, 1991], each theoretically proven to be of equivalent efficiency. It employs Vickery's Mechanism [Vickery 1961], where the highest bid wins; however, the winning bidder has to pay the second-highest bid price. This is the sealed-bid analog to an English auction: a bidder must only beat the next highest bidder to win; therefore it may be willing to pay more than its last bid. In contrast to most negotiation protocols considered so far in MAS research, the selection of the highest bid for a partial process is a multi-attribute problem (i.e., the individual bidding enterprises must present a 7-tuple, describing for example the enterprise's deviation of planned sales from actual sales, to the service manager, representing their bid for a partial process).

4.4.3 Competitive Negotiation in Telecommunications

The negotiation techniques employed in the telecommunications domain generally involve simple bidding mechanisms based on the contract net (recall 4.3.2). Richer constraint relaxation for bargaining power is required and is currently under research in these projects.

- *Network Management [Skarmeas & Clark, 1996]*

  In this multi-agent application, agents are controlling a switch-based network in order to provide support to customer requests for services of different requirements in network resources. Each agent is decomposed into simpler processes, one such process being responsible for coordination and negotiation. Using negotiation, the agents handle (1) customer requests such as the routing of a call from Los Angeles to London and (2) network node failure by deciding who will be the new manager of resources controlled by a recently failed agent.

  Communication and coordination occurs using a central message-board, which has been implemented using the language platform, April (Agent Process Interaction Language). Hence, coordination adopts the master/slave approach. Advertisements for services (e.g., routing) are posted to the message-board, which then forwards (or broadcasts) them onto the relevant parties. Bidders respond directly to the sender (not through the message board), who then selects the best offer.

- *Service Management*

  Telecommunication networks are growing in number and complexity. Customers wish to be provided with customised and diverse services using a variety of terminals and connected to the network of their choice. Therefore, despite their diversity, competitive service delivery platforms are forced to interconnect. Distributed Agent Technology provides a candidate solution for the required intelligent service management. One such system, which attempts to apply agent technology to the problem of service management in telecommunications is that designed by Marius Busuioc (1996).

  Customers send service requests to their individual Customer Agents (CA), who then attempts to configure the service by sending a request to a number of relevant service agents (SA). These service agents negotiate with service supplier agents (SSA) to select those with the highest bids that can provide the service content which can best be contracted to satisfy the CA's original service request.

4.4.4 Competitive Negotiation Using Human-Inspired Approaches

It appears that almost every form of human interaction requires some degree of explicit or implicit negotiation [Werkman, 1990]. Hence, it is not surprising that many negotiation researchers draw from human negotiation strategies. One such system is PERSUADER [Sycara, 1989], which adopts a case-based approach to negotiation, based on Sycara's belief that human negotiators draw from the past negotiation experiences to guide present and future ones. The system consists of three agents: a company, its union, and the mediator whose task is to help the other two agents reach an acceptable compromise. PERSUADER's input is the set of conflicting goals of the company and union, and the dispute context. Its final output is either a single plan in the form of an agreed upon compromise, or an indication of failure of the parties to the dispute did not reach agreement in a particular number of negotiation cycles. Its high level tasks are:-

- propose an initial compromise
- repair and improve a rejected compromise
- persuade the parties to change their evaluations of a compromise (using past negotiation experiences).

4.4.5 Cooperative Negotiation

The negotiation strategies described below can be applied to CMAS, namely, where the individual agents are cooperative and will collaborate in order to achieve a common goal and for the best interests of the system as a whole.

- ***Cooperative Negotiation Using A Cyclic Model***
  Bussmann and Muller's negotiating framework for cooperating agents [Bussmann & Muller, 1992] involves a cyclic negotiation model which is both general and simple. They attempt to address many of the limitations of other negotiation proposals/models. The cyclic nature of the model addresses the thorny issue of conflict resolution.

  The general strategy is that negotiation begins with one, some or every agent making a proposal. Then agents evaluate and check the proposals against their preferences, and criticise them by listing their preferences which are violated by the proposals. The agents then update their knowledge about other agents' preferences and the negotiation cycle resumes with a new proposal or proposals in the light of the newly gleaned information. Conflicts between agents are handled in a concurrent conflict resolution cycle.

- ***Collaborative Planning for Air Traffic Control***
  Chu-Carroll and Carberry (1995) have developed a cooperative negotiation model for the air traffic control domain, in which the participating agents (pilots, air traffic control) collaborate on developing the best plan in terms of the interests of the agents as a group. This model attempts to solve the following scenario:

  "The key air-traffic control systems in the country of Ruritania suddenly fail, due to freak weather conditions. Fortunately, computerised air-traffic control systems in neighbouring countries negotiate between themselves to track and deal with all affected flights, and the potential disastrous situation passes without major incident"

  This model involves an agent detecting conflicts regarding proposed actions and beliefs from other agents and initiates collaborative negotiation to resolve such conflicts. This negotiation involves the agent modifying the proposal with appropriate justification for this modification, based on its own beliefs. In this way, agents collaborate to achieve mutual beliefs, thereby resolving conflicts. Like Sycara's PERSUADER system above, these researchers also adopt the idea that *evidence* improves the persuasiveness of a message/belief.

- ***Using Mentalistic Notions for Air Traffic Control***
  Rao and Georgeff (1995) introduce collaborative agents with much stronger definitions, namely, they characterise agents in terms of the mentalistic notions, Belief, Desire and Intention. These are attitudes typical of epistemic logics. A representative BDI architecture (the PRS [Georgeff & Lansky, 1987] is illustrated in Figure 4.2, depicting how a BDI architecture contains four key data structures:
    - An agent's *beliefs*, correspond to information the agent has about the world, which may be incomplete or incorrect;
    - An agent's *desires* intuitively correspond to the tasks allocated to it (its goals);
    - An agent's *intentions* represent desires that it has committed to achieving. The intuition is that an agent will not, in general, be able to achieve *all* its desires, even if these desires are

consistent. Agents must therefore fix upon some subset of available desires and commit resources to achieving them. These chosen desires, which the agent has committed to achieving, are *intentions*. An agent will typically continue to try and achieve an intention until either it believes the intention is satisfied, or else it believes the intention is no longer achievable.

- The final data structure in a BDI agent is a *plan library*, which is a set of plans (or recipes) which specify courses of actions that may be followed by an agent in order to achieve its intentions.

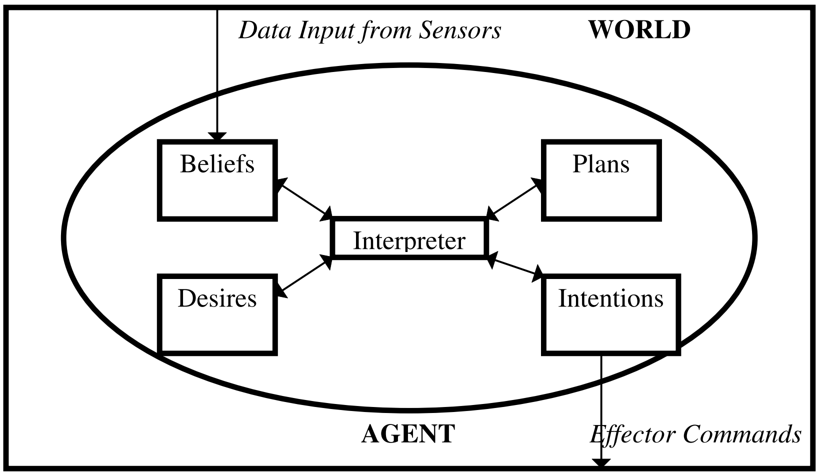


Figure 4.2 *A BDI Agent Architecture*

The *interpreter* in Figure 4.2 is responsible for updating beliefs from observations made of the world, generating new desires (tasks) on the basis of new beliefs, and selecting from the set of currently active desires some subset to act as intentions. Finally, the interpreter must select an action to perform on the basis of the agent's current intentions and procedural knowledge.

This BDI agent framework is currently undergoing parallel evaluation trials at Sydney airport, receiving live data from the radar. Essentially, each aircraft agent estimates its Estimated Time of Arrival (intentions), based on its beliefs (e.g., current wind velocity) and desires (desired ETA) and using accessible world semantics.

Other agent frameworks based on mentalistic notions include Bratman et al's (1988) IRMA and Jennings' (1993) GRATE/GRATE* environments. Much of the latter work exploits Cohen and Levesque's classic (1990) classic work.

4.5 Communication in MAS Research

The above sections have briefly explored some key techniques and applications to achieve coordination in a multi-agent environment. In order to achieve this coordination, the agents **might** have to negotiate, however, they **must** interact and exchange information; therefore they need to ***communicate***. The importance of agents in a MAS 'understanding' a communication language is emphasised by Genesereth and Ketchpel (1994), who go as far to suggest that "an entity is an agent of and only if it communicates correctly in an Agent Communication Language" (ACL). Although this statement is arguable because it fails to mention other attributes that are considered germane to agenthood, the importance of having an ACL is acknowledged - imagine a society without some lingua franca. ACLs derive their inspiration from *speech act theory* [Searle, 1969], which was developed by linguists in an attempt to understand how humans use language in every day situations, to achieve everyday tasks, such as requests, orders, promises, etc.

In multi-agent systems, the possible solutions to the communication problem range from those involving no communication, those involving ad hoc ACLs to those involving standard ACLs:-

- ***No Communication or Primitive Communication***

In some cases an agent rationally infers other agents' plans *without communicating* with them. Tubbs (1984) term this *tacit bargaining* and point out that it works best when agents' goals are not conflicting. To study this behaviour, Rosenschein (recall section 4.4.1) used a game-theoretic approach characterised by payoff matrices that contain agents' payoffs for each possible outcome of an interaction. These payoffs are common knowledge to all agents and once an interaction has been recognised, there is no further communication among the agents; each must decide on its action alone using the payoff matrix.

Primitive Communication is restricted to some finite set of fixed signals (usually two) with fixed interpretations (Hoare, 1978) and has been applied by Georgeff (recall section 4.3.3) in his mult-agent planning approach to coordination. The limited number and types of signals available in this approach limit the coordination between agents. Hence, requests, commands, and complex intentions cannot be expressed using these signals.

- ***Ad Hoc ACLs***
  To date, many multi-agent system applications employ an *ad hoc* set of performatives within *ad hoc* agent communication languages. Many others, strictly speaking, do not have explicit ACLs - they communicate by depositing information in some shared data structure. Naturally, the shortcoming of such *ad hoc* approaches is that it makes it non-trivial, if not impossible, for interoperation to occur between agent applications built by different developers.

  - ***Plan Passing and Message Passing***
    In the plan *passing approach*, an agent A1 communicates its total plan to agent A2 and A2 then communicates its total plan back to A1. Whichever plan arrives first is accepted. This method of communication has not been used much as it presents severe limitations [Rosenschein, 1986] including the fact that total plan passing is computationally expensive.

    The work by Hewitt and his colleages (kornfeld and Hewitt, 1981) on actor languages is a significant application of *message passing* in DAI. An actor can be defined as "a computational agent which carries out its actions in response to processing a communication. The actions it may perform are: send communication to itself or to other actors; create more actors;etc.", [Agha and Hewitt, 1988]. Several other works in DAI have used classic message passing with a protocol and a precise content, such as the Contract Net (recall section 4.3.2).

  - ***Information Exchanges through a Blackboard***
    In AI, the *blackboard* is the model of shared memory that is most often used. It is a repository on which agents write messages, post partial results, and obtain information. Section 4.3.1 indicated several applications which employ this communication scheme to achieve coordination.

- ***Standard ACLs***
  The case for having an ACL standard is not only compelling, but also essential. The FIPA (Foundation for Intelligent Physical Agents) standards organisation are attempting to find such standards. The main two standards proposed to date are KQML and ARCOL. Based on a meeting in January in Turin, FIPA are now tending towards ARCOL, although the issue is still under active discussion.

  The Knowledge Query and Manipulation Language is an evolving standard ACL, being developed as part of the DARPA knowledge-sharing effort (KSE). The KQML language can be viewed as consisting of three layers - the content, message and communication layers: the content layer specifies the actual content of the message; the set of performatives provided by the language constitute the message layer (e.g., tell, reply, ask-if, advertise, etc.); the protocol for delivering the message that subsumes the content defines the communication layer.

  KQML has a number of shortcomings, the most important of which is its lack of precise *semantics*. Cohen and Levesque (1995) believe this to be a real problem: "without a precise semantics, agent designers cannot be certain that the interpretation they are giving to a 'performative' is in fact the same as the one some other designer intended to have" . ARCOL, which has been developed in France Telecom by Sadek et al (1996), attempts to overcome these limitations by incorporating a logic of mental attitudes and hence semantics into the message level. Furthermore, the number of performatives are reduced. However, both ACLs still present a problem, namely, there is no level of semantics at the communication level, so that if agent A sends a request to agent B, who dies, how is Agent A meant to react ? This obviously presents a problem with real world, open and heterogeneous multi-agent systems.

4.6 Conclusion

This section has provided the reader with an overview of key approaches to Distributed Problem Solving (coordination, cooperation, negotiation and communication) in Multi-Agent systems. Successful coordination is a key design objective for most multi-agent system builders. Without good coordination mechanisms, many of the benefits of the multi-agent paradigm disappear. The various approaches discussed in this paper have their relative advantages and disadvantages and hence at this time, there is no universally best method. There seem to be two extremes : the theoretical methods produce good results in well-constrained environments, but many of their underpinning assumptions are not well suited to developing real world systems. On the other hand, the less formal techniques operate well in limited domains, but suffer from a lack of grounding and rigorous evaluation.

Currently, most research assumes agents are cooperative. However, the issue of ***trust*** is of paramount importance as consumers will not buy untrustworthy agents systems. Self-Interested agents may be programmed by different parties that clearly have their own goals in mind and hence might act insincerely. Research in trust usually uses tools based on Rosenschein's game theory [URL17] extends game theoretic and other microeconomic techniques to the context of multiple computational agents. Game theoretic techniques allow one to build multiagent systems that cannot be manipulated by agents acting insincerely.

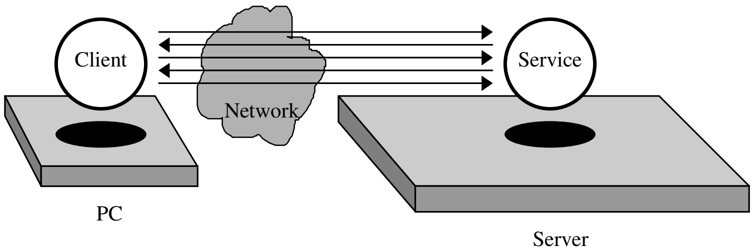
5. Intelligent Mobile Agents

In this chapter we examine Intelligent Mobile Agent technology. In examining this topic we will provide a description of the basic technology, an analysis of various mobile agent implementations, and finally we will examine some applications to which mobile agent technology has been applied.

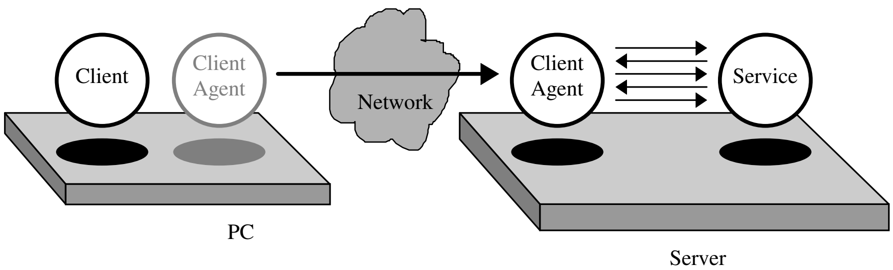
5.1 A working definition

We begin with a rather loose definition [White, 1995]. A mobile agent is a program that can migrate from machine to machine in a heterogeneous network. The program chooses when and where to migrate. It can suspend its execution at an arbitrary point, transport itself to another machine and resume execution.

In the first picture off Figure 5.1 below a client communicates with a remote service over a network using the client-server paradigm. However in the second picture the client creates a mobile agent to do its bidding, then dispatches it to the remote service where it can interact locally with that service.



Old RPC-based client-server computing paradigm.



New mobile agent computing paradigm.

Figure 5.1 *Difference between RPC paradigm and the Mobile Agent approach.*

Mobile agent technology originally emerged from advances made in distributed systems research [Cardelli, 1995]. They are also known as itinerant agents [Chess et al, 1995a; Chess et al, 1995b], transportable agents [Kotz et al, 1996] and re-locatable objects [Joseph et al, 1995]. The idea of transporting programs around a network is not a new one. Batch processing can be considered as a primitive form of mobile agent technology, which has been around since the 1960's. However, we believe that the basic ability of a program to transport itself to new locations is not sufficient to describe a mobile agent.

5.2 A more refined definition

Based on existing literature [Chess et al, 1995a; White, 1995; Nwana, 1996; URL1] we believe that the following definitions sufficiently characterise the essence of a mobile agent system.

A mobile agent is a software entity which exists in a software environment. It inherits some of the characteristics of an Agent (as defined in chapter 2). A mobile agent must contain all of the following models: an agent model, a life-cycle model, a computational model, a security model, a communication model and finally a navigation model.

This defines a mobile agent. However we must also consider the software environment in which mobile agents exist. We call this the mobile agent environment. The following is a definition of a mobile agent environment:

A mobile agent environment is a software system which is distributed over a network of heterogeneous computers. Its primary task is to provide an environment in which mobile agents can execute. The mobile agent environment implements the majority of the models which appear in the mobile agent definition. It may also provide: support services which relate to the mobile agent environment itself, support services pertaining to the environments on which the mobile agent environment is built, services to support access to other mobile agent systems, and finally support for openness when accessing non-agent-based software environments.

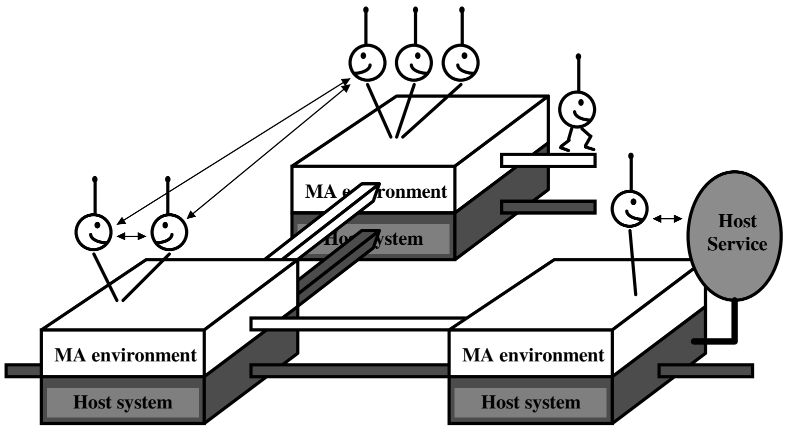


Figure 5.2 *Basic mobile agent architecture*

The above definitions state the essence of a mobile agent and the environment in which it exists. A mobile agent system is illustrated in Figure 5.2. The mobile agent environment is built on top of a host system. The smiley faces are mobile agents which travel between mobile agent environments. Communication between mobile agents (local and remote) is represented by bi-directional arrows. Communication can also takes place between a mobile agent and a host service. In the following sections we will examine the importance of each of the models in a mobile agent.

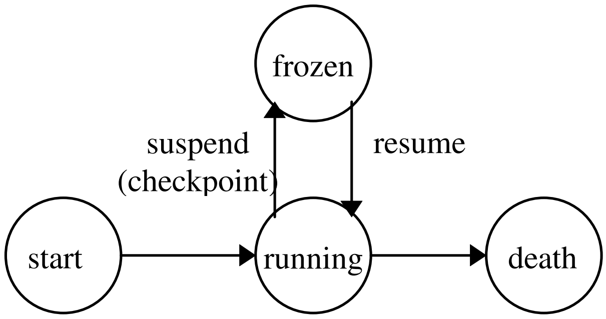
5.2.1 Agent model

This model defines the internal structure of the intelligent agent part of a mobile agent [URL1; Nwana, 1996]. The structure of this model was presented in chapter 2. In essence it defines the autonomy, learning and co-operative characteristics of an agent. Additionally, it specifies the reactive and proactive nature of agents.

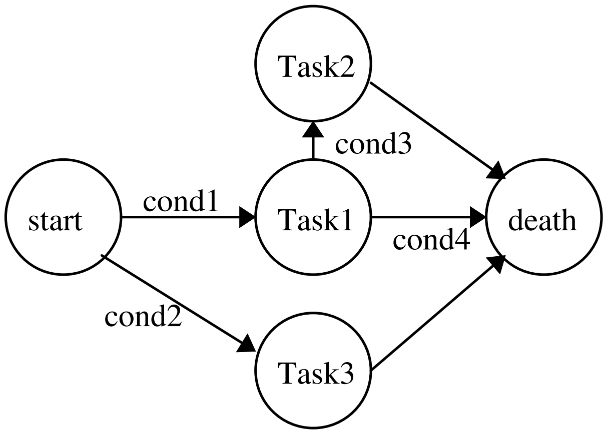
5.2.2 Life-cycle model

This model defines the different execution states of a mobile agent and the events that cause the movement from one state to another [URL1]. Thus, it is closely related to the computational model which describes *how* the execution occurs. Today, the most prominent life cycle models are the persistent process model, adopted by Telescript [White, 1995] and AgentTCL [Kotz et al, 1996], and the task based model, adopted by Aglets [Chang and Lange, 1996].

The first, and more powerful life cycle (first diagram of Figure 5.3), starts with a ‘start’ state, proceeds to a ‘running’ state where a persistent process is executed, and hopefully enters a ‘death’ state where the process is terminated. When a mobile agent is transported from one node to another, the process in the running state is check-pointed [White, 1995] and the agent enters the ‘frozen’ state. Next, its context is delivered to the destination node where the process is resumed and re-enters the ‘running’ state at the point it left off. This kind of life cycle is the most powerful and flexible life cycle [Kotz et al, 1996] as all other life cycle models can be built on top of this model.



Persistent process based life cycle (Telescript, AgentTCL).



Task based life cycle (Aglets).

Figure 5.3 *Task based and persistent process based life cycles.*

The task based model (second diagram of Figure 5.3) starts in the 'start' state. Then depending on a set of conditions is proceeds through a network of tasks. Each task having its own state. However, when the agent moves to a new node the context of the currently executing task is lost. Before moving the agent must indicate the first task to be started when it re-materialises on the destination node. The flexibility of this approach is reduced by the lose of context information during transport [Mira da Silva and Rodrigues da Silva, 1997].

5.2.3 Computational model

The computational model defines how a mobile agent executes when it is in a 'running' state [URL1; Ousterhout, 1994; Lindholm et al, 1996]. The computation takes place in an environment and is facilitated by some form of processor. A processor could be the CPU of the a computer or a more abstract processor as can be found in the Java virtual machine [Lindholm et al, 1996].

As part of this model a set of primitive instructions must be specified. This defines the computational abilities of an agent. These include data manipulation instructions and thread control instructions.

Implementers of a mobile agent gain access to the other models through the computational model, hence the structure of the computational model effects all other models. If we have a very general and powerful computational model we will have to consider its effect on the life-cycle, security, communication and navigational aspects of a mobile agent.

5.2.4 Security model

Mobile agent security can be split into two broad areas [Chess, 1996]. The first involves the protection of host nodes from destructive mobile agents while the second involves the protection of mobile agents from destructive hosts.

Protection of hosts from malicious agents

A mobile agent system is an open system [Chess et al, 1995a; Chess, 1996; Tardo and Valenta, 1996]. Therefore, just like in any open system, the host nodes are subject to a variety of attacks, both old and new. Attacks on host security fall into four main categories [Coulouris et al, 1994]:

- *Leakage*: acquisition of data by an unauthorised party.
- *Tampering*: alteration of data by an unauthorised party.
- *Resource stealing*: use of facilities by an unauthorised party.
- *Vandalism*: malicious interference with a hosts data or facilities with no clear profit to the perpetrator.

The traditional methods of attack include eavesdropping, masquerading, message tampering, message replay and viruses. A mobile agent can employ any of these methods of attack, which in turn, can be guarded against using standard techniques such as cryptography, authentication, digital signatures and trust hierarchies [Coulouris et al, 1994].

However a mobile agent is unique in that its code is executed by a host. Thus an executing mobile agent has automatic access to some of a hosts resources. With this level of access mobile agents can mount attacks by altering other local agents, propagating viruses, worms and Trojan horses, impersonating other users and mounting denial of service attacks [Chess, 1996; Tardo and Valente, 1996]. The standard approach to this problem is to reject all unknown code from entry into a host. This is not a viable solution in a mobile agent environment. Telescript [White, 1995] offers one approach to the problem.

The Telescript approach [White, 1996] provides three mechanisms which can be applied at various degrees of granularity.

- *Process (Agent) Safety and Security*: Allows the safe interaction among agents and between agents and the host. It achieves this by using Authenticated Identities, Protected References, quotas on Permits and Engine-mediated Protocols for agent rendezvous with other agents and host entry.
- *System Safety*: controls access to system supplied resources. This is achieved by forcing access to these resources through what are effectively proxy objects. The following proxies are supported: External File (file access), External Handle (network access) and the Control Manager (management functions).
- *Network Security*: provides authentication facilities, communication privacy and system level authorisation. This is achieved through the use of Region Policies, Secure Channels and Export Restrictions.

They key to this approach is the fact the Telescript is based on an interpreted language which facilitates detailed control over the capabilities of the unknown process/agents running on top of it. Both Java [Gosling and McGilton, 1995] and Safe-Tcl [Borenstein, 1994] also provide similar security features. Finally, there is a fine line between a secure system and an unusable one. Therefore is important to chose carefully the trade-offs between flexibility and security [Chess, 1996]

Protection of agents from malicious hosts

This, the second area, deals with the issue of protecting mobile agents from hosts which want to scan the agent for information, alter the agents state or code, or kill the agent. The critical problem is that in order for the agent to run it must expose its data and code to the host environment which supplies the means for that agent to run [Chess, 1996]. Thus the mobile agent is unprotected from the host.

Current consensus is that it is *computationally impossible* to protect a mobile agent from a malicious host [Rasmusson and Janson, 1996a]. Instead of tackling the problem from a computational (hard) point of view, current research [Rasmusson and Janson, 1996a; URL9] is looking at sociological (soft) means of enforcing good host behaviour.

In conclusion, although the protection of hosts can be achieved by using conventional and recently introduced mechanisms, a large question remains over the security of the mobile agent.

5.2.5 Communication model

Mobile agents would be little use if they were unable to communicate with other entities in a computing environment. These entities include users, other agents (static or mobile), the host mobile agent environment and other systems such as CORBA [OMG, 1996] based distributed systems.

Communication is used when accessing services outside [OMG, 1996] of the mobile agent, during co-operation and co-ordination [URL9; URL1; Finin & Wiederhold, 1991] between mobile agents and other entities, and finally to facilitate competitive behaviour between self-interested agents [URL9].

A protocol is an implementation of a communication model. There exists a wide variety of protocols which range from the most basic, (email and RPC) to the more general and complex (KQML). Protocols also vary depending on the types of communicating entities. Humans are unlikely to want to communicate with an agent using a KQML protocol, they would prefer human speech or interaction through a modern GUI system.

Because of the range of protocols used during communication it is probable that mobile agents will require more than one communication model. This is equivalent to the multilingual abilities of humans.

With numerous communication protocols comes the need for translation [URL9] between protocols. Such translation abilities might be internal to a mobile agent or could be contracted from an external translation service. This is similar to the way ministers in the European Union communicate through human translators.

5.2.6 Navigation model

Finally we take a look at the navigation (mobility) aspect of mobile agents. This model concerns itself with all aspects of agent mobility from the discovery and resolution [White, 1995] of destination hosts to the manner in which a mobile agent is transported. This part of mobile agents is receiving the greatest amount of attention at the present time [White, 1995; Chang & Lange, 1996; Kotz et al, 1996]. Most researchers in this area agree that the following are required:

- Naming conventions for all entities in the mobile agent system (agents, hosts, services etc.).
- Access to information regarding a remote mobile agent environment.
- The ability to move a mobile agent into a ‘suspended’ life-cycle state ready for transportation to a remote host.
- The ability to transport a mobile agent, which is in the ‘suspended’ state, to a remote mobile agent environment.
- The ability to receive a suspended mobile agent from a remote host and reconstitute it in a new environment.

Some optional services which aid navigation:

- Directory and referential services can aid mobile agents in the discovery of relevant services which the agent might visit. Current systems include X.500 [Rose, 1992] and CORBA Trader. New and up coming systems are LDAP [URL8] and FIPA [URL1].
- Network topology information services could supply information about the state of the network. This information can help a mobile agent to make planning decisions based on the quality of different parts of the underlying network(s).

5.2.7 Summary

Although each of the models above have been individually addressed, they are in fact highly integrated and interdependent. Figure 5.4 is an illustration of the structure of a simple mobile agent. The core is based on the computational model. This has significant impact on the other models. It defines how we address other agents, hosts and resources which is important to the security model. The type of life cycle model adopted is dependent on the facilities of the computational model. Thus all current mobile agent implementations based on the Java [Gosling and McGilton, 1995] (Aglets [Chang and Lange, 1996] being one such system) language have been forced to adopt the task based model. However the computational model of AgentTcl [Kotz et al, 1996] and Telescript [White, 1995] is such that they support a persistent process life cycle model.

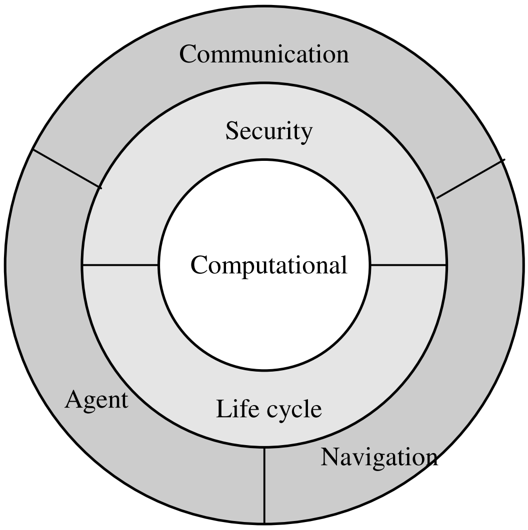


Figure 5.4 *Simple view of the structure of a mobile agent.*

Both the security and life cycle models are structurally very close to the core. Security issues permeate every aspect of a mobile agent and therefore must be provided for at the most basic level. In Telescript primitive security features are implemented as authenticated identities, protected (read only) references, permits (a form of capability), secure channels and engine-mediated protocols for receiving new agents and agent meeting. The life cycle model defines the valid states for an agent.

The outer layer contains the communication, navigation and agent models. The agent model defines the 'intelligent agent' aspects of a mobile agent such as learning and collaboration functions. The communication model is heavily dependent on the security model so that agents are not corrupted by other agents or hosts. Finally the navigation model is also dependent on the security model as it hands itself over to the host to be transported to another node. The transportation process often moves the agent into a different life cycle state as in Aglets [Chang and Lange, 1996].

5.3 Proposed advantages of mobile agent technology

In this section we examine various claims related to mobile agents and its advantages over pre-existing computing paradigms such as client-server computing [Chess et al, 1995a; Chess et al, 1995b; Kotz et al, 1996]. In general we have not discovered any client-server function which is uniquely enabled by mobile agent technology. For almost every mobile agent-based function proposed, we could propose an alternative based on an existing technology [Chess et al, 1995a]. However we do believe that in certain cases mobile agents have advantages over conventional approaches, at the design, implementation and execution stages. We examine some of the proposed advantages:

- **Efficiency —** Mobile agents consume fewer network resources since they move the computation to the data rather than the data to the computation [URL3].
- **Reduction of network traffic —** Most communication protocols involve several interactions, especially when security measures are enabled. This causes a lot of network traffic. With mobile agents one can package up a conversation and ship it to a destination host where the interactions can take place locally [Chess et al, 1995b; Hurst et al, 1997].
- **Asynchronous autonomous interaction —** Tasks can be encoded into mobile agents and then dispatched. The mobile agent can operate asynchronously and independent of the sending program [Joseph et al, 1995; Hurst et al, 1997]. An example of this would be a mobile device dispatching an *autonomous* search agent onto the fixed network, disconnecting, then reconnecting some time later to collect the results of the search.
- **Interaction with real-time entities —** Real-time entities such as software controlling an ATM switch or a safety system in a nuclear installation require immediate responses to changes in their environment. Controlling these entities from across a potentially large network will incur significant latencies. For critical situations (nuclear system control) such latencies are intolerable. Mobile agents offer an alternative. They can be dispatched from a central system to control real-time entities at a local level and also process directives from the central controller [Krause, 1997; URL4].
- **Dynamic adaptation —** Related to the above topic, mobile agents have the ability to autonomously react to changes in their environment. However such changes must be communicated to mobile agents from the mobile agent environment [Krause, 1997; Hurst et al, 1997].
- **Dealing with vast volumes of data —** When vast volumes of data are stored at remote locations, as in weather information systems, the processing of this data should be performed local to the data, instead of transmitting it over a network [URL3].
- **Robustness and fault tolerance —** The ability of mobile agents to react dynamically to adverse situations makes it easier to build fault tolerant behaviour, especially in a highly distributed system [URL4; Hurst et al, 1997].
- **Support for heterogeneous environments —** Both the computers and networks on which a mobile agent system is built are heterogeneous in character. As mobile agent systems are generally computer and network independent, they support transparent operation [Chang & Lange, 1996; Kotz et al, 1996; Cardelli, 1995] .
- **Personalise server behaviour —** In the Intelligent Network domain mobile agents are proposed as a way to personalise the behaviour of network entities (e.g. routers) by dynamically supplying new behaviour [URL4].
- **Support for electronic commerce —** Mobile agents can be used to build electronic markets. Here mobile agents embody the intentions, desires, and resources of the participants in such a market [White, 1995; URL5].
- **Convenient development paradigm —** The design and construction of distributed systems can be made easier by the use of mobile agents. Mobile agents are inherently distributed in nature and hence are a natural view of a distributed system [Cardelli, 1995; Chang & Lange, 1996].

5.3.1 Summary

There are many alternatives to mobile agents such as queued RPC, proxy servers and client-server computing. The problem with these alternative techniques is that each one is only suitable for certain applications[Chess et al, 1995a]. A mobile agent system on the other hand is a single, unified framework in which a wide range of distributed applications can be implemented efficiently and easily.

5.4 Mobile agent implementations

This section of the review concerns itself with the current state-of-the-art in mobile agent systems. It reviews a couple of enabling technologies on which a mobile agent system is built, and then proceeds to review three of the most important mobile agent systems in use today.

5.4.1 Introduction

Mobile agent technology has been under serious development since about 1994. However it is only in the last six months that useful mobile agent systems have appeared. Most of today's mobile agent systems are built on top of the Java system [Gosling & McGilton, 1995] or the Tcl/tk system [Ousterhout, 1994]. However some projects have attempted to build mobile agent systems from the ground up. Telescript [White, 1995] is one such system. Below we present a table of the most important mobile agent systems:

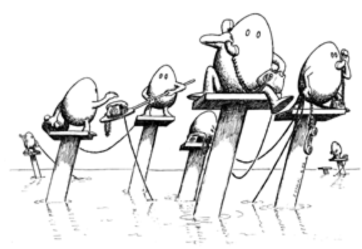
Table 5.1 List of important mobile agent systems.

System	Based-on	Status	Organisation
Java	enabling	Version 1.1	Sun Microsystems.
Tcl/tk	enabling	Version 8.0a2	Sun Microsystems
Obliq	enabling	Research only	Digital
Aglets	Java	Alpha4	IBM Japan
AgentTcl	Tcl/tk	Alpha1.1	Dartmouth College
Telescript	custom	Too old and costly	General Magic, Inc.
Mole [Baumann, 1996]	Java	Too old	University of Stuggart
Ara [URL6]	Tcl	Too old	University of Kaiserslautern,
Tacoma	Tcl	Research only	Cornell University and the University of Tromso

Java, Tcl/tk and Obliq are systems which enable the building of mobile agent frameworks. We will review Obliq and Java. The remaining entries are mobile agent systems. We will only review the most important of these, Aglets, AgentTcl, Telescript and Ara.

5.4.2 Obliq

Obliq [Cardelli, 1995] is a lexically-scoped un-typed interpreted language that supports distributed object-oriented computation. An Obliq computation may involve multiple threads of control within an address space, multiple address spaces on a machine, heterogeneous machines over a local network, and multiple networks over the Internet.



Obliq objects have state (data) and are local to a site. Obliq computations can roam over the network, while maintaining network connections. This ability to roam satisfies a major requirement of mobile agent technology.

Obliq is one of the oldest mobile agent systems around. It is impractical as an implementation system. However it has proved to be a very useful for prototyping the mobility algorithms of mobile agents. Similar languages include Python [URL10] and April [McCabe & Clark, 1994].

5.4.3 Java

The Java environment [Gosling & McGilton, 1995] is a new approach to distributed computing. It is an interpreted, object-oriented language and library set. Its main features are:

- Simple and familiar object-oriented language.
- Architecture neutral, portable and robust system.
- Interpreted and dynamic program execution.
- A comprehensive, but not perfect, security system.
- Multithreaded execution with synchronisation between threads.
- Sophisticated and comprehensive networking capabilities.
- Support for distributed systems through the Remote object invocation (RMI) and Object serialisation (OS) facilities.

Taken individually, the characteristics discussed above can be found in a variety of software development environments. What's completely new is the manner in which Java and its run-time system have combined them to produce a flexible and powerful programming system which supports distributed computing.

Java, on the other hand, contains no mobile agents. It is simply an enabling technology for mobile agents. In this role it has recently received a lot of attention which has resulted in significant revisions of the important RMI and OS facilities.

5.4.4 Aglets

The aglet system [Chang & Lange, 1996] represents the next leap forward in the evolution of executable content on the Internet, introducing program code that can be transported along with state information. Aglets are Java objects that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it brings along its program code as well as its data. Conceptually, the aglet is a mobile agent because it supports the ideas of autonomous execution and dynamic routing on its itinerary. An aglet can also leverage all other facilities of the Java system.

Aglets is a comprehensive mobile agent system with the following features:

1. A globally unique naming scheme for agents (navigation/security model).
2. A travel itinerary for specifying complex travel patterns with multiple destinations and automatic failure handling (navigation model).
3. A white board mechanism allowing multiple agents to collaborate and share information asynchronously (communication model).
4. An agent message-passing scheme that supports loosely coupled asynchronous as well as synchronous peer-to-peer communication between agents (communication model).
5. A network agent class loader that allows an agent's Java byte code and state information to travel across the network, and (navigation model).
6. An execution context that provides agents with a uniform environment independent of the actual computer system on which they are executing (computational model).

Unfortunately the life-cycle model of an Aglet agent is quite simple. Only the state is maintained as an aglet travels from host to host. It would be preferable if the current execution context of an aglet could be transported as well as the state. Research in Cambridge University, England, should make this possible with Java within the next six months.

Aglets are possibly the most applicable mobile agent technology at the present moment in time.

5.4.5 AgentTcl

AgentTcl [Kotz et al, 1996] is a simple platform independent mobile agent system. Currently it lacks the same degree of sophistication that the Aglets system has achieved. It is built on top of the Tcl/tk scripting system and lacks the power and flexibility of a full computer language such as Java.

The agent model is simply a Tcl script, hence its computational model is based on the Tcl/tk scripting language. The navigation model is based around a single command `agent_jump`. This command can appear anywhere in an agent and will cause all its state and execution context to be frozen and transported to a specified host. This ability to transport the execution context gives Tcl agents a more sophisticated life-cycle model than Aglets. However the mobile agent environment does not provide any navigation support services.

Three commands, `agent_send`, `agent_receive` and `agent_meet`, make up the AgentTcl communication model. It is a simple system allowing for asynchronous and stream based communication. PGP encryption and the Safe-Tcl module are used as the basis for a security model. The security is quite basic and is relatively untested at this stage as AgentTcl has had little exposure outside of Dartmouth College.

5.4.6 Telescript

Telescript [White, 1995] is an object-oriented mobile agent language. It claims to be *"a platform that enables the creation of active, distributed network applications"*. Its main achievements to date are its involvement in the Sony Magic Link project, its Magic Cap product, and its research into electronic market places. Telescript has three main concepts: agents, places and the 'go' instruction. Agents travel from place to place using the 'go' instruction. They can interact with each other and any services located at the places through the use of the 'meet' instruction.

Telescript is a whole new platform-independent system consisting of a language, and interpreter called the Telescript engine. The places in Telescript are equivalent to our concept of a mobile agent environment where static services are located at a host. The computational model is the Telescript language itself and the agent model is simply a Telescript object. A Telescript agent maintains its execution state during travel resulting in a flexible life-cycle model.

Special instructions implement part of the other models. The navigation model is implemented by the concept of places and the 'go' instruction. The 'meet' and 'connect' instructions are the basis of the communication model. Finally the security model is implemented with the authority, region and identity concepts. Access to resources is granted through the use of permits.

Although Telescript is technically a very sophisticated mobile agent system it is also costly. A Telescript engine requires significant computer resources to run (e.g. 96 Mbytes of memory). Financially it is a very expensive piece of software which inhibits its acceptance as a general mobile agent system among nearly all mobile agent developers. This contrasts strongly with the Java & Aglets and Tcl/tk & AgentTcl combinations which are both free. However, it should be noted that General Magic are making changes to their system. They are also involved in the Agent Society [URL9] and have made a CORBA standards submission.

5.4.7 Ara

Ara [Peine and Stolpmann, 1997] is another platform for mobile agents in heterogeneous networks. Ara perceives a mobile agent as program with the "ability to change its host machine during execution while preserving its internal state". Hence in preserving the internal state of a mobile agent, heavy emphasis is placed on security issues.

Ara aims to provide comprehensive mobile agent functionality while at the same time leveraging as much as possible of the established programming models and languages.

In order to leverage current programming models Ara provides a core which exists above an unmodified host operating system. This core is then connected to a modified interpreter of the destination language in order to provide mobile agent facilities to programs of that language. It should be noted that the mobility and security functions are provided in an orthogonal fashion and do not interfere with the normal execution of programs. Currently interpreters of the following programming languages have adapted to the Ara system: C, C++, Java and Tcl.

Ara provides mobility at the program level resulting in large, course grained mobile agents. This type of mobile agent is not suitable in many circumstances such as in mobile computing and communication.

5.4.8 Summary

The above systems represent the current state of the art in mobile agent technology. Although the Telescript system greatly advanced the design of mobile agent technology, it has been deserted in favour of cheaper alternatives such as Aglets. To Aglets advantage they have adopted the now well known Java system as their implementation language.

It should be noted that these systems are only a first cut. There are still many unsolved issues.

- Security is possibly the single greatest issue, especially the protection of mobile agents themselves. Most current work concentrates on the security of the host [White, 1995]. However Ara [Peine and Stolpmann, 1997] is investigating mobile agent partial protection through the use of authentication certificates. Whereas Ara is approaching the issue from a computational point of view, the Swedish Institute of Computer Science [Rasmusson and Janson, 1996a; Rasmusson and Janson, 1996b] and the Agent Society [URL9] believe computational protection of a mobile agent to be fundamentally impossible. They advocate sociologically enforced protection of a mobile agent.

- Each alternative implementation of a mobile agent system supports its own communication mechanism. Java based systems (Aglets) often use the standard method invocation mechanism. Telescript has its own meeting and channel communication system. What about communication between agents in *different* agent systems? Ara does support a form of inter-agent system communication based on a common Ara core. Any system supporting access to agent via CORBA should, in principle, be able to communicate with any other mobile agent systems which supports CORBA.

- The life cycle model of the leading mobile agent system (Aglets) is task based. This ties its users into a single model which reduces their flexibility to implement their own system/application specific life cycle models. The work in this thesis is a case in point. There is a recent technological push [Mira da Silva and Rodrigues da Silva, 1997] for a more powerful model (persistent process model) which will allow greater flexibility in creating specialised mobile agents..

- Although the transportation mechanism necessary for agent mobility is in place, more work is required on facilitating resource discovery and the patterns of agent movement. The OMG have recently requested submissions for a Mobile Agent Facility [Chang and Covaci, 1997] which it is hoped would leverage of the plethora of existing CORBA Services [OMG, 1994].

In conclusion, it is apparent that mobile agent development is still in its infancy. It is only now that the first practical systems are emerging from research. Aglets would seem to be the most practical system available at the moment, but this is likely to change if the OMG's Mobile Agent Facility reaches fruition.

5.5 Applications of mobile agents

In this section of the review we shall examine some applications of mobile agent technology. We will also study why mobile agent technology was chosen in preference to other competing technologies.

5.5.1 Introduction

Only recently have mobile agent systems become sufficiently well developed to consider using them as tools in other projects. Because of this, the majority of projects which use mobile agent technology are only in a research phase.

The following table is a representative list of projects which use mobile agent technology. The majority of projects are either telecommunications or mobile computing and communications based. This reflects the current belief that mobile agent technology is best applied to these areas.

Table 5.2 *Representative sample of mobile agent projects in industry and academia*

System	Mobile agent technology	Organisation
InAMoS	custom	TU Berlin
Rover	custom	MIT
IBM - AMP	custom	IBM
Magic Cap	Telescript	General Magic
Magna	CORBA and Java	GMD Fokus
Stormcast	Tacoma	Cornell University and the University of Tromsø
Perpetuum Mobile Procura Project	custom (Java)	Carleton University

We summarise each system in the following sections.

5.5.2 InAMoS

The InAMoS project [URL5] attempts to bring electronic market places to the mobile user. Like Telescript, they propose the use of a mobile agent to represent a user in this market place. They make a distinction between two types of agent. The first is a mobile user agent which represents the human consumers in this market place. Secondly, ordinary mobile agents and services.

5.5.3 Rover: Mobile application toolkit

"The Rover toolkit [Joseph et al, 1995] combines re-locatable dynamic objects and queued remote procedure calls to provide unique services for "roving" mobile applications". A re-locatable dynamic object is a mobile agent which can be exchanged among clients and servers in order to reduce the communication requirements in a very hostile environment. Queued remote procedure calls form the communication model that permits mobile agents to interact with other agents, and services, either locally or remotely situated.

5.5.4 IBM Agent Meeting Point for Mobile Communication

This work concerns itself with the creation of a framework for mobile agents which are to be used to implement secure, remote applications in large public networks with many mobile computing devices such as laptops and PDAs. Key to this idea is the design of an Agent Meeting Point (AMP)

[Chess et al, 1995b] where mobile agents can interact with each other, and support services. It is believed that the mobile agents offer the following advantages in such a system:

- Reduction of overall communication traffic over very low bandwidth wireless links.
- Ability to engage in high-bandwidth communication with servers when accessing remote services.
- Ability to interact with many big systems without detailed knowledge of their capabilities ahead of time.
- Personalising services by having agents take up residence at a server and provide alternative interfaces to that server.

5.5.5 Magic Cap

Magic Cap [URL7] is a near total intelligent personal communications system. It uses the mobile agent technology in Telescript to allow different forms of communication (phone, fax, email, etc.) to intelligently interact with the user irrespective of his geographical location.

5.5.6 Magna

GMD Fokus and the Technical University of Berlin have recognised the possible benefits that mobile agent technology may inject into telecommunications, particularly with respect to the service scalability problem inherent in Intelligent Networks. They have, however, also realised that the value of intelligent mobile agents for the provision of telecommunication services may be questionable for some application domains in which traditional RPC mechanisms might be more adequate. Therefore, they are currently developing a mobile agent architecture (MAGNA) [Krause, 1997], which considers Remote Programming and Remote Procedure Call as potentially *complementary* and not mutually exclusive technologies. It is their belief that this technology will enable telecommunication services to be provided instantly and customised directly at the locations where the intelligence is needed, namely it will enable 'Intelligence on Demand'.

5.5.7 Stormcast

Stormcast [URL3] is project that has been running for many years. It provides critical weather information to the meteorological office in Norway regarding weather conditions in Norway and the significant expanse of sea of its west coast.

To increase access to the meteorological data they have decided to leverage mobile agent technology [Johansen et al, 1995]. These mobile agents are created at the client side (in a WWW browser) and sent to servers which can take their content and retrieve the necessary information from the vast meteorological data stores.

This is an example of how mobile agent technology is used to process huge amounts data located at remote hosts.

5.5.8 Perpetuum Mobile Procura Project

This project [URL4] concerns itself with advanced network management through the use of mobile agent technology. They believe that mobile agent technology can be used to a) overcome the problems of legacy systems, b) explore new intelligent distributed management techniques with mobile agents and finally c) deliver a Java-based platform for both real and simulated network management.

Mobile agents in this system are called Netlets. It is believed that these netlets will support the following capabilities in a mobile agent enabled network management system:

- service by delegation,
- installable/extensible/modifiable services,
- fault management (diagnosis/recovery),

- plug-and-play networks (auto-provisioning),
- self-repairing network,
- interface to legacy systems,
- network security.

5.5.9 Summary

In summary, it is evident that significant attention is being paid to the application of mobile agent technology in the areas of telecommunications and mobile computing and communications.

5.6 Summary

In this chapter, on Intelligent Mobile Agents, we have examined the requirements of mobile agent technology, reviewed some early mobile agent implementations, and finally we have surveyed a selection of telecommunication based applications.

It has become evident from the past twelve months of research that mobile agent technology is not a passing fad. Rather it is set to accelerate as more companies invest in it and the technology matures.

Given that mobile agent technology has been warmly embraced by developers, and given that they believe it holds much promise, we believe that this technology should have a key role in telecommunication systems of the future.

6. Summary

This review surveyed the diverse literature on agent technology, and more specifically within the three research areas of Intelligent User Interfaces, Distributed Agent Technology and Mobile Agent Technology. We will now summarise the relevant chapters in turn:

Chapter 3 presents an overview of the existing approaches to the creation of Intelligent Interface Agents. We examine some of the possible technologies for creating one of the key components of such systems namely user models and present a possible way forward in this area. We conclude by pointing out the relevance of this area to telecoms companies in their role as Internet Service Providers.

Chapter 4 presented an overview of the existing key approaches to Distributed Problem Solving (coordination, cooperation, negotiation and communication) in Multi-Agent systems and their diverse applications. Successful coordination is a key design objective for most multi-agent system builders. Without good coordination mechanisms, many of the benefits of the multi-agent paradigm disappear. There seem to be two extremes: the theoretical methods produce good results in well-constrained environments, but many of their underpinning assumptions are not well suited to developing real world systems. On the other hand, the less formal techniques operate well in limited domains, but suffer from a lack of grounding and rigorous evaluation. Currently, most research assumes agents are cooperative. However, the issue of ***trust*** is of paramount importance as consumers will not buy untrustworthy agents systems. This issue still needs to be tackled effectively in this area.

In chapter 5, on Intelligent Mobile Agents, we have examined the requirements of mobile agent technology, reviewed some early mobile agent implementations, and finally we have surveyed a selection of telecommunication based applications. It has become evident from the past twelve months of research that mobile agent technology is not a passing fad. Rather it is set to accelerate as more companies invest in it and the technology matures. Given that mobile agent technology has been warmly embraced by developers, and given that they believe it holds much promise, we believe that this technology should have a key role in telecommunication systems of the future.

In essence, the overall message of this review is that agents are here to stay, not least because of their diversity, their wide range of applicability and broad spectrum of companies investing in them. As we move further and further into the information age, any information-based organisation which does not invest in agent technology may be committing commercial hara-kiri.

7. References

[Agha, G. & Hewitt, 1988]
Agha, G. and Hewitt, C., *Concurrent programming using actors*, Object-Oriented Concurrent Programming (Y.Younezawa and M. Tokoro, eds.), MIT Press, Cambridge, MA, (1988).

[Balabanovic 1997].
Balabanovic M. An Adaptive Web Page Recommendation Service. In Proceedings of Autonomous Agents 1997 Marina del Rey CA, USA

[Bates 1992]
Bates J. The Nature of Characters in Interactive worlds and The Oz Project. Virtual Realities: Anthology of industry and Culture, Ed. C E Loeffler, 1993.

[Baumann, 1996]
Baumann J., M. Straher and F. Hohl,. *Mole–A Java based mobile agent system*. In Proceedings of the 2nd ECOOP Workshop on Mobile Object Systems, July 1996, Linz, Austria.

[Bond & Gasser, 1988]
Bond and Gasser, *Readership in Distributed Artificial Intelligence,* Morgan Kaufmann, San Mateo, CA, (1988)

[Bratman et al, 1988]
Bratman, M. Israel, D. and Pollack, M., *Plans and Resource-Bounded Practical Reasoning*, Computational Intelligence 4, pages 349-355, (1988).

[Bussmann & Muller, 1992]
Bussmann, S. and Muller, J., *A Negotiation Framework for Co-operating Agents,* in Dean S M (Ed): 'Proc CKBS-SIG', Dake Centre, University of Keele, pages 1-17, (1992).

[Busuioc, 1996]
Busuioc, M., *Distributed Intelligent Agents - A Solution for the Management of Complex Services,* The Intelligent Agents for Telecom Applications (ECAI '96) Workshop Proceedings, (1996).

[Cammarata et al, 1993]
Cammarata, S., McArthur, D., Steeb, R., *Strategies of Cooperation in Distributed Problem Solving*, Proceedings of the Eight International Joint Conference on Artificial Intelligence, pages 767-770, (1983).

[Cardelli, 1995]
Cardelli L., *A language with distributed scope*. In Computing Systems, 8(1):27-59, 1995.

[Carpenter and Grossberg 1995]
Carpenter A and Grossberg S. Adaptive Resonance Theory(ART) . In the handbook of Brain Theory and Neural Networks. Ed. M. Arbib. Pub. MIT Press 1995.

[Chaib-draa, 1996]
Chaib-draa, B., *Connection between Micro and Macro Aspects of Agent Modelling*, Proceedings of the First International Conference on Autonomous Agents, (1997).

[Chang & Lange, 1996]
Chang D. and D. Lange, *Mobile Agents: A New Paradigm for Distributed Object Computing on the WWW*. 1996, OOPSLA'96 Workshop.

[Charlton et al]
Charlton P. et al An Open Agent Architecture for Integrating Multimedia Services. In Proceedings of AA '97 Marina Del Rey California.

[Chavez & Maes, 1996]
Chavez, A. and Maes, P., *Kasbah: An Agent Marketplace for Buying and Selling Goods,* Proceedings of The First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, (1996).

[Chess et al, 1995a]
Chess D., C. Harrison and A. Kershenbaum, *Mobile Agents: Are they a good idea.* Technical Report, March 1995, IBM T.J. Watson Research Center, NY.

[Chess et al, 1995b]
Chess D., B. Grosof, C. Harrison, D. Levine, C. Parris and G. Tsudik, *Itinerant Agents for Mobile Computing.* Technical Report, October 1995, IBM T.J. Watson Research Center, NY.

[Chu-Carroll & Carberry, 1995]
Chu-Carroll, J. and Carberry, S., *Conflict Detection and Resolution in Collaborative Planning*, Intelligent Agents II, Lecture Notes in Artificial Intelligent 1037, Heidelberg: Springer Verlag, (1995).

[Cohen & Levesque, 1990]
Cohen, P. and Levesque, H., *Intention is Choice with Commitment*, Artificial Intelligence 42, pages 213-261, (1990).

[Cohen & Levesque, 1995]
Cohen, P. and Levesque, H., *Communicative Actions for Artificial Agents,* Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, Cambridge, AAAI Press, pages 65-72, (1995).

[Conry et al, 1986]
Conry, S., Meyer, R. and Lesser, V., *Multistage Negotiation in Distributed Planning*, COINS Technical Report, University of Massachusetts, Amherst, Boston, (1986).

[Corkill, 1979]
Corkill, D., *Hierarchical Planning in a Distributed Environment*, Proceedings of the Sixth IJCAI, Cambridge, MA, Morgan Kaufmann, San Mateo, CA, pages 168-175, (1979).

[Davies et al 1996].
Davies N.J., Weeks R and Revett M.C. *Information Agents for the World Wide Web. BT Technical Journal* 14:4 October 1996.

[Davis & Smith, 1983]
Davis, R. and Smith, R., *Negotiation as a Metaphor for Distributed Problem Solving*, Artificial Intelligence 20, pages 63-109, (1983).

[Decker, 1987]
Decker, K.S., *Distributed Problem-Solving Techniques: A Survey*, IEEE Trans. Syst. Man. Cybernet 17(5), pages 729-740, (1987).

[Doorenbos et al, 1996]
Doorenbos, R., Etzioni, O. and Weld, D., *A Scaleable Comparison-shopping agents for the World Wide Web*, Technical Report UW-CSE-96-01-03, University of Washington, 1996.

[Durfee et al, 1987]
Durfee, E-H, Lesser, V. and Corkill, D., *Coherent Cooperation among Communicating Problem Solvers,* IEEE Transactions of Computers C-36(ii), pages 1275-1291, (1987).

[Durfee et al, 1989]
Durfee, E., Lesser, V. and Corkill, D., *Trends in Cooperative Distributed Problem Solving*, IEEE Knowledge & Data Engineering, 1, No. 1, pages 63-83, (1989).

[Esfandiari et al 1996]

Babak Esfandiari, Deflandre G, and Quinqueton J. *An interface agent for network supervision*. In Proceedings of ECAI 96 Budapest.

[Etzioni & Weld 1994]
Etzioni O. & Weld D. A softbot-based Interface to the Internet. Communications of the ACM 37:7 72-76 1994.

[Farmer et al, 1996]
Farmer W., J Guttman and V. Swarup, *Security for mobile agents: Issues and requirements*. In Proceedings of the 19th National Information Systems Security Conference, October 1996, Baltimore, USA.

[Finin & Wiederhold, 1991]
Finin T. and G. Wiederhold, *An Overview of KQML*. 1991, Dept. of Computer Science, Stanford University, USA.

[Fischer et al, 1996]
Fischer, K., Muller, J., Hemig, I. and Sheer, A-W, *Intelligent Agents in Virtual Enterprises*, Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, (1986).

[Foner 1997].
Foner L. YENTA: *A Multi-Agent, Referral-Based Matchmaking System*. In Proceedings of Autonomous Agents 1997, Marina del Rey CA,USA.

[Galbraith, 1977]
Galbraith, J.R., *Organisation design*, Addison-Wesley, (1977).
[Garcia-Molina & Yan 1994] H. Garcia-Molina & Yan T. 1994. *Index Filtering For Information Filtering Under the Vector Space Model* ICDE '94.

[Genesereth & Ketchpel, 1994]
Genesereth, M. and Ketchpel, S., *Software Agents*, Communications of the ACM, 37, No. 7, pages 48-53, (1994).

[Genesereth, 1986]
Genesereth, M., Ginsberg, M. and Rosenchein, J., *Cooperation without Communication*, Proc. Annual Assoc. Artificial Intelligence, Philadelphia, pages 51-57, (1986).

[Georgeff & Lansky, 1987]
Georgeff, M. and Lansky, A., *Reactive reasoning and planning,* in Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87), pages 677-682, Seattle, WA, 1987.

[Georgeff, 1983]
Georgeff, M., *Communication and Interaction in Multi-Agent Planning,* In Proceedings of the Third National Conference on Artificial Intelligence, Washington, D.C., Morgan-Kaufmann, San Mateo, California, pages 125-129, (1983).

[Georgeff, 1984]
Georgeff, M., *A Theory of Action for Multi Agent Planning*, Proceedings of the Fourth National Conference on Artificial Intelligence, Austin, Texas, San Mateo, California, pages 121-125, (1984).

[Gosling & McGilton, 1995]
Gosling J. and H. McGilton, *The Java Language Environment: A White Paper*. Sun Microsystems, 1995.

[Guilfoyle, 1995]
Guilfoyle, C., *Vendors of Agent Technology, UNICOM Seminar on Intelligent Agents and their Business Applications*, 8-9 November, London, 135-142.

[Hewitt, 1977]
Hewitt, C., *Viewing Control Structures as Patterns of Passing Messages*, Artificial Intelligence, 8, No. 3, pp 323-364 (1977).

[Hurst et al, 1997]
Hurst L., P. Cunningham and F Somers, *Mobile agents–Smart messages*. To appear in the Proceedings of the 1st International Workshop on Mobile Agents, April 1997, Berlin, Germany.

[Jennings, 1993]
Jennings, N., *Specification and Implementation of a Belief Desire Joint-Intention Architecture for Collaborative Problem Solving,* Journal of Intelligent and Cooperative Information Systems 2(3), pages 289-318, (1993).

[Johansen et al, 1995]
Johansen D., R. van Renesse and F. Schneider, *An introduction to the TACOMA distributed system*. Technical Report 95-23, Dept. of Computer Science, University of Tromsø, Tromsø, Norway.

[Joseph et al, 1995]
Joseph D., A. deLespinasse, J. Tauber, D. Gifford and M. Frans Kaashoek, *Rover: A toolkit for mobile information access*. In the proceedings of the 15th ACM Symp. On Operating Systems Principles, December 1995.

[Kearney et al, 1994]
Kearney, P., Sehmi, A. and Smith, R., *Emergent behaviour in a multi-agent economics simulation,* Cohn A G (Ed): 'Proceedings of the 11th European Conference on Artificial Intelligence', London, John Wiley, (1994).

[Kornfeld & Hewitt, 1981]
Kornfeld, W. and Hewitt, C., *The Scientific Community Metaphor*, IEEE Trans. Syst. Man, Cybernet: SMS-11(1), pages 24-33, (1981).

[Kotz et al, 1996]
Kotz D., R. Gray and D. Rus, *Transportable Agents Support Worldwide Applications.* In Proceedings of the 7th ACM SIGOPS European Workshop, September 1996, Connemara, Ireland.

[Krause, 1997] Krause S., *MAGNA-A DPE-based Platform for Mobile Agents in Electronic Service Markets*. Submission for ISADA '97 - The Third International Symposium on Autonomous Decentralised Systems, 9-11 April 1997, Berlin, Germany

[Lashkari et al]
Y. Lashkari, M. Metral, P. Maes, *Collaborative Interface Agents*, Proceedings of the Twelfth National Conference on Artificial Intelligence, Vol. 1, AAAI Press, Seattle, WA, August 1994.

[Lesser & Corkill, 1981]
Lesser, V. and Corkill, D., *Functionally accurate, cooperative distributed systems*, IEEE Transactions on Systems, Man and Cybernetics, 11, No.1, pages 81-96, (1981).

[Lesser & Corkill, 1983]
Lesser, N. and Corkill, D., *The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks,* AI Magazine, 4, No. 3, pages 15-33, (1983).

[Lester et al 1997]
Lester J.C., FitzGerald B.A. & Stone B.A. *The Pedagogical Design Studio: Exploiting Artifact-Based Task Models for Constructivist Learning*. In Proceedings of IUI 1997 Orlando, Florida.

[Lindholm & Yellin, 1996]
Lindholm T. and F. Yellin, *The Java Virtual Machine Specification.* Addison-Wesley, 1996.

[Luce & Raiffa, 1957] Luce, R. and Raiffa, H., *Games and Decisions*, John Wiley & Sons, (1957).

[Maes 1997]
P. Maes. Intelligent Software. In Proceedings of Intelligent User Interfaces 1997, Orlando Florida.

[Mazur 1994]
J. Mazur. *Learning and Behaviour*. Ed. J. Mazur , Pub Prentice Hall 1994

[McCabe & Clark, 1994]
McCabe F. and K Clark*, April: Agent Process Interaction Language.* November 1994.

[Montelius et al, 1996]
Montelius, J., Janson, S. and Gabrielsson, J, *Intentions and Intelligent Screening in an Agent-Based Personal Communication System,* In Intelligent Agents for Telecoms Applications, ECAI '96 Workshop Proceedings.

[Moukas & Zacharia 1997].
Moukas A. and G. Zacharia. *Evolving a Multi-Agent Information Filtering Solution in Amalthaea*. In Proceedings of Autonomous Agents 1997 Marina del Rey CA,USA

[Murnighan, 1991]
Murnighan, J., *The Dynamics of Bargaining Games*, Englewood Cliffs, NJ: Prentice Hall, (1991).

[Nwana et al, 1996]
Nwana, H., Lee, L. and Jennings, N., *Coordination in software agent systems*, BT Laboratories internal report, (1994).

[Nwana, 1996]
Nwana H., *Software agents: An Overview*. Knowledge and Engineering Review, 11(3) November 1996.

[O'Hare & Jennings, 1996]
O'Hare, G. and Jennings, N. (Eds), *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, 1996.

[Odubuyi et al 1997].
Odubiyi J. et al. SAIRE - *A Scalable Agent-based Information Retrieval Engine*. In Proceedings of Autonomous Agents 1997 Marina del Rey CA, USA.

[OMG, 1996]
OMG, *CORBA 2.0 specification*, ptc/96-03-04.

[Ousterhout, 1994]
Ousterhout J., *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[Pandit and Kalbag 1997]
Pandit M. and Kalbag S. *The Selection Recognition Agent : Instant Access to Relevant Information and Operations*. In Proceedings of Intelligent User Interfaces 1997, Orlando Florida.

[Parunak, 1987]
Parunak, H., *Manufacturing Experience with the Contract Net,* In Gasser L. and Huhns M. (Eds): 'Distributed Artificial Intelligence 2', Morgan Kaufmann (1989)

[Quinlan 92]
J. R. Quinlan, 1992. *C4.5: Programs for machine learning*. California: Morgan Kaufmann.

[Ramusson & Janson, 1996a]
Rasmusson A. and S. Janson, *Personal security assistance for secure Internet commerce*. In New Security Paradigms '96, ACM Press, Sept 1996.

[Ramusson & Janson, 1996b]

Rasmusson L. and S. Janson, *Simulated social control for secure Internet commerce*. In New Security Paradigms '96, ACM Press, Sept 1996.

[Rao & Georgeff, 1995]
Rao, A. and Georgeff, M., *BDI Agents: From Theory to Practice,* Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, (June 1995).

[Ritter 1995]
Ritter H. *Self-Organising Feature Maps*. In Brain Theory and Neural Networks Ed. M. Arbib. Pub MIT Press 1995

[Rose, 1992]
Rose M., *The Little Black Book (Mail Bonding with OSI Directory Services)*. Prentice Hall, 1992.

[Rosenchein, 1986]
Rosenchein, J., *Rational interactions: Cooperating among intelligent agents*, Ph.D. Dissertation, Computer Science Department, Stanford University, Stanford, CA, (1986).

[Rosenschein & Genesereth, 1985]
Rosenschein, J. and Genesereth, M., *Deals among Rational Agents,* Proceeding Intelligent Agents Conference Artificial Intelligence 9th, Los Angeles, pages 91-99, (1985).

[Rosenschein & Zlotkin, 1994]
Rosenchein, J. and Zlotkin, G., *Rules of Encounter Designing Contentions for Automated Negotiation among Computers,* MIT Press (1994).

[Rumelhart et al 1986]
D. E. Rumelhart, G. E. Hinton and R. J. Williams*, Learning Internal Representation by Error Propagation* in the book of D. E. Rumelhart and J. L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volume 1, Bradford Books, pages 318--362, 1986.

[Scheiderman 1997]
Scheiderman B, *Direct Manipulation for Comprehensible, Predictable and Controllable User Interfaces*. In Proceedings of Intelligent User Interfaces 1997, Orlando Florida.

[Searle, 1969]
Searle, J., *Speech Acts*, Cambridge, MA, Cambridge University Press, (1969).

[Skarmeas & Clark, 1996]
Skarmeas, N. and Clark, K., *Process Oriented Programming for Agent-Based Network Management,* The Intelligent Agents for Telecoms Applications (ECAI ‘96) Workshop Proceedings, (1996).

[Smith & Davis, 1981]
Smith, R. and Davis, R., *Frameworks for Cooperation in Distributed Problem Solving*, IEEE Transactions on Systems, MAN and Cybernetics, Bol. SMC-11, No. 1, (Jan 1981).

[Sycara, 1989]
Sycara, K., *Multi-Agent Compromise via Negotiation*, Gasser, L. and Huhns, M. (Eds): ‘Distributed Artificial Intelligence 2’, Morgan Kaufmann, (1989).

[Thomas and Fischer 1997]
Thomas C. & Fischer G. Using Agents to Personalise the Web. In Proceedings of Intelligent User Interfaces 1997, Orlando Florida

[Thompson et al, 1991]
Thompson, G., Frances, J., Levacic, R. and Mitchell, J. (Eds), *Market, Hierarchies and Networks - the Coordination of Social Life*, Sage Publications, (!991).

[Titmuss et al, 1996]

Titmuss, R. , Crabtree, I.B., and Winter, C.S., *Agents, Mobility and Mulitmedia Information,* In BT Technology Journal, Vol 14 No 4 October 1996

[Tsvetovatyy & Gini, 1996]
Tsvetovatyy, B. and Gini, M., *Toward a Virtual Marketplace,* Proceedings of The First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, (1996).

[Tubbs, 1984]
Tubbs, S. (Ed), *A System Approach to Small Group Interaction*, 2nd ed. Addison-Wesley, Reading, MA, (1994).

[URL1]
http:/drogo.cselt.stet.it/fipa/

[URL2]
Chess D*., Things that go bump in the night.* http://www.research.ibm.com/massive/bump.html.

[URL3]
http://www.cs.uit.no/DOS/StormCast/

[URL4]
http://www.sce.carleton.ca/netmanage/perpetuum.shtml

[URL5]
http://www.cs.tuberlin.de/ cs/research/english/projects/inamos.html

[URL6]
http://www.uni-kl.de/AG-Nehmer/Ara/ara.html

[URL7]
http://www.genmagic.com/MagicCap/

[URL8]
http://www.umich.edu/~rsug/ldap/ldap.html

[URL9]
http://www.agent.org/

[URL10]
http://www.python.org/

[URL11]
Bargain Finder Web Page URL :- http://bf.cstar.ac.com/bf

[URL12]
Web Hound Web Page URL :- http://webhound.www.media.mit.edu/projects/webhound

[URL13]
Crayon Web Page URL :- http://www.crayon.net.

[URL14].
Altavista Search Engine Web Page http://www.altavista.digital.com

[URL15].
Infoseek Search Engine Web Page http://www.infoseek.com

[URL16].
Yahoo Web Directory Starting Point http://www.yahoo.com

[URL17]
http://dis.cs.umass.edu/research/negotiation.html

[URL18]
http://drogo.cselt.stet.it/fipa/torino/cfp1/proposit-013.htm

[URL19]
Sen, S. *"Using Formal Specification to Resolve Conflicts between Contracting Agents"*
http://euler.mcs.utulsa.edu/~sandip/DAI.html

[URL20]
http://www.centigram.com/centigram/Products/Technology/Truvoice/TruVoice-Brochure.html

[Vergera 1994].
Vergara H. PROTUM: A Prolog-based tool for User Modelling . University of Konstanz, D-78434, Konstanz, Germany 1994.

[Vickery, 1961]
Vickery, W., *"Counterspeculation, auctions and competitive sealed tenders",* Journal of Finance, 16, pages 8-37, (1961).

[Watkins 89]
Watkins, C., 1989.*Learning from Delayed Rewards*,Thesis,University of Cambidge,England.

[Werkman, 1990]
Werkman, K., *"Knowledge-based model of negotiation using shareable perspectives"*, Proceedings of the 10th International Workshop on DAI, Texas, (1990).

[Werner 1996]
Co-operating Agents: A Unified Theory of Communication and Social Structure. In Distributed Artificial Intelligence Vol 2. Ed. Gasser and Huhns, Pub. Morgan Kaufmann

[White, 1995]
White J., *Telescript technology: The foundation of the electronic market place*. General Magic white paper 1995.

[Wittig 1992].
Wittig T. (editor) *ARCHON: An Architecture for Multi-agent Systems.* Ellis Horwood, Publisher 1992