11

Agents have found application in many domains: in this chapter, I will describe some of the most notable. Broadly speaking, applications of agents can be divided into two main groups.

Distributed systems. In which agents become processing nodes in a distributed system. The emphasis in such systems is on the 'multi' aspect of multiagent systems.

Personal software assistants. In which agents play the role of proactive assistants to users working with some application. The emphasis here is on 'individual' agents.

.1 Agents for Workflow and Business Process Management

Workflow and business process control systems art an area of increasing importance in computer science. Workflow systems aim to automate the processes of a business, ensuring that different business tasks are expedited by the appropriate people at the right time, typically ensuring that a particular document flow is maintained and managed within an organization. The ADEPT system is a current example of an agent-based business process management system (Jennings *et al.*, 199Gb). In ADEPT, a business organization is modelled as a society of negotiating, service providing agents.

More specifically, the process was providing customers with a quote for installing a network to deliver a particular type of telecommunications service. This activity involves the following British Telecom (BT) departments: the Customer Service Division (CSD), the Design Division (DD), the Surveyor Department (SD), the Legal Division (LD), and the various organizations which provide the outsourced service of vetting customers (VCs). The process is initiated by a customer contacting the CSD with a set of requirements. In parallel to capluring the requirements, the CSD gets the customer vetted. If the customer fails the vetting procedure, the quote process terminates. Assuming the customer is satisfactory, their requirements are mapped against the service portfolio. If they can be met by an off-the-shelf item, then an immediate quote can be offered. In the case of bespoke services, however, the process is more complex. CSD further analyses the customer's requirements and whilst this is occurring LD checks the legality of the proposed service. If the desired service is illegal, the quote process terminates. If the requested service is legal, the design phase can start. To prepare a network design it is usually necessary to dispatch a surveyor to the customer's premises so that a detailed plan of the existing equipment can be produced. On completion of the network design and costing, DD informs CSD of the quote. CSD, in turn, informs the customer. The business process then terminates.

From this high-level system description, a number of autonomous problem-solving entities were identified. Thus, each department became an agent, and each individual within a department became an agent. To achieve their individual objectives, agents needed to interact with one another. In this case, all interactions took the form of negotiations about which services the agents would provide to one another and under what terms and conditions. The nature of these negotiations varied, depending on the context and the prevailing circumstances: interactions between BT internal agents were more cooperative than those involving external organizations, and negotiations where time is plentiful differed from those where time is short. In this context, negotiation involved generating a series of proposals and counter-proposals. If negotiation was successful, it resulted in a mutually agreeable contract. The agents were arranged in various organizational structures: collections of agents were grouped together as a single conceptual unit (e.g. the individual designers and lawyers in DD and LD, respectively), authority relationships (e.g. the DD agent is the manager of the SD agent), peers within the same organization (e.g. the CSD, LD, and DD agents) and customer-subcontractor relationships (e.g. the CSD agent and the various VCs).

The ADEPT application had a clear rationale for adopting an agent-based solution. Centralized workflow systems are simply too unresponsive and are unable to cope with unpredictable events. It was decided, therefore, to devolve responsibility for managing the business process to software entities that could respond more rapidly to changing circumstances. Since there will inevitably be inter-dependencies between the various devolved functions, these software entities must interact to resolve their conflicts. Such a method of approach leaves autonomous agents as the most natural means of modelling the solution. Further arguments in favour of an agent-based solution are that agents provide a software model that is ideally suited to the devolved nature of the proposed business management system. Thus, the project's goal was to devise an agent framework that could be used to build agents for business process management. Note that ADEPT was neither conceived nor implemented as a general-purpose agent framework.

Rather than reimplementing communications from first principles, ADEPT was built on top of a commercial CORBA platform (OMG, 2001). This platform provided the basis of handling distribution and heterogeneity in the ADEPT system. ADEPT agents also required the ability to undertake context-dependent reasoning and so a widely used expert system shell was incorporated into the agent architecture for this purpose. Development of either of these components from scratch would have consumed large amounts of project resources and would probably have resulted in a less robust and reliable solution. On the negative side, ADEPT failed to exploit any of the available standards for agent communication languages. This is a shortcoming that restricts the interoperation of the ADEPT system. In the same way that ADEPT exploited an off-the-shelf communications framework, so it used an architecture that had been developed in two previous projects (GRATE* (Jennings, 1993b) and ARCHON (Jennings *et al.*, 1996a)). This meant the analysis and design phases could be shortened since an architecture (together with its specification) had already been devised.

The business process domain has a large number of legacy components (especially databases and scheduling software). In this case, these were generally wrapped up as resources or tasks within particular agents.

ADEPT agents embodied comparatively small amounts of AI technology. For example, planning was handled by having partial plans stored in a plan library (in the style of the Procedural Reasoning System (Georgeff and Lansky, 1987)).

main areas in which AI techniques were used was in the way agents negotiated with one another and the way that agents responded to their environment. In the former case, each agent had a rich set of rules governing which negotiation strategy it should adopt in which circumstances, how it should respond to incoming negotiation proposals, and when it should change its negotiation strategy. In the latter case, agents were required to respond to unanticipated events in a dynamic and uncertain environment. To achieve their goals in such circumstances they needed to be flexible about their individual and their social behaviour.

ADEPT agents were relatively coarse grained in nature. They represented organizations, departments or individuals. Each such agent had a number of resources under its control, and was capable of a range of problem-solving behaviours. This led to a system design in which there were typically less than 10 agents at each level of abstraction and in which primitive agents were still capable of fulfilling some high-level goals.

11.2 Agents for Distributed Sensing

The classic application of multiagent technology was in distributed sensing (Lesser and Erman, 1980; Durfee, 1988). The broad idea is to have a system constructed as a network of spatially distributed sensors. The sensors may, for example, be acoustic sensors on a battlefield, or radars distributed across some airspace. The global goal of the system is to monitor and track all vehicles that pass within range of the sensors. This task can be made simpler if the sensor nodes in the network *cooperate* with one another, for example by exchanging predictions about when a vehicle will pass from the region of one sensor to the region of another. This apparently simple domain has yielded surprising richness as an environment for experimentation into multiagent systems: Lesser's well-known *Distributed Vehicle Monitoring Testbed* (DVMT) provided the proving ground for many of today's multiagent system development techniques (Lesser and Erman, 1980).

11.3 Agents for Information Retrieval and Management

The widespread provision of distributed, semi-structured information resources such as the World Wide Web obviously presents enormous potential; but it also presents a number of difficulties (such as 'information overload'). Agents have widely been proposed as a solution to these problems. An *information agent* is an agent that has access to at least one and potentially many information sources, and is able to collate and manipulate information obtained from these sources in order to answer queries posed by users and other information agents (the network of interoperating information sources are often referred to as intelligent and cooperative information systems (Papazoglou *et al.*, 1992)). The information sources may be of many types, including, for example, traditional databases as well as other information agents. Finding a solution to a query might involve an agent accessing information sources over a network. A typical scenario is that of a user who has heard about somebody at Stanford who has proposed something called agent-oriented programming. The agent is asked to investigate, and, after a careful search of various Web sites, returns with an appropriate technical report, as well as the name and contact details of the researcher involved.

To see how agents can help in this task, consider the Web. What makes the Web so effective is that

- it allows access to networked, widely distributed information resources;
- it provides a uniform interface to multi-media resources including text, images, sound, video, and so on;

- it is hypertext based, making it possible to link documents together in novel or interesting ways; and
- perhaps most importantly, it has an extraordinarily simple and intuitive user interface, which can be understood and used in seconds.

The reality of Web use at the beginning of the 21st century is, however, still somewhat beset by problems. These problems may be divided into two categories: *human* and *organizational*

Human factors

The most obvious difficulty from the point of view of human users of the World-Wide Web is the 'information overload' problem (Maes, 1994a). People get overwhelmed by the sheer amount of information available, making it hard for them to filter out the junk and irrelevancies and focus on what is important, and also to actively search for the right information. Search engines such as Google and Yahoo attempt to alleviate this problem by indexing largely unstructured and unmanaged information on the Web. While these tools are useful, they tend to lack functionality: most search engines provide only simple search features, not tailored to a user's particular demands. In addition, current search engine functionality is directed at textual (typically HTML) content - despite the fact that one of the main selling features of the Web is its support for heterogeneous, multi-media content. Finally, it is not at all certain that the brute-force indexing techniques used by current search engines will scale to the size of the Internet in the next century. So *finding* and *managing* information on the Internet is, despite tools such as Google, still a problem.

In addition, people easily get bored or confused while browsing the Web. The hypertext nature of the Web, while making it easy to link related documents together, can also be disorienting - the 'back' and 'forward' buttons provided by most browsers are better suited to linear structures than the highly connected graph-like structures that underpin the Web. This can make it hard to understand the topology of a collection of linked Web pages; indeed, such structures are inherently difficult for humans to visualize and comprehend. In short, it is all too easy to become lost in cyberspace. When searching for a particular item of information, it is also easy for people to either miss or misunderstand things.

Finally, the Web was not really designed to be used in a methodical way. Most Web pages attempt to be attractive and highly animated, in the hope that people will find them interesting. But there is some tension between the goal of making a Web page animated and diverting and the goal of conveying information. Of course, it *is* possible for a well-designed Web page to effectively convey information, but, sadly, most Web pages emphasize appearance, rather than content. It is telling that the process of using the Web is known as 'browsing' rather than 'reading'. Browsing is a useful activity in many circumstances, but is not generally appropriate when attempting to answer a complex, important query.

Organizational factors

In addition, there are many organizational factors that make the Web difficult to use. Perhaps most importantly, apart from the (very broad) HTML standard, there are no standards for how a Web page should look.

Another problem is the cost of providing online content. Unless significant information owners can see that they are making money from the provision of their content, they will simply cease to provide it. How this money is to be made is probably the dominant issue in the development of the Web today. I stress that these are not criticisms of the Web - its designers could hardly have anticipated the uses to which it would be put, nor that they were developing one of the most important computer systems to date. But these are all obstacles that need to be overcome if the potential of the Internet/Web is to be realized. The obvious question is then: what more do we need?

In order to realize the potential of the Internet, and overcome the limitations discussed above, it has been argued that we need tools that (Durfee *et al.*, 1997)

- give a single coherent view of distributed, heterogeneous information resources;
- give rich, *personalized,* ~~user-oriented~~ services, in order to overcome the 'information overload' problem - they must enable users to find information they really want to find, and shield them from information they do not want;
- are scalable, distributed, and modular, to support the expected growth of the Internet and Web;
- are adaptive and self-optimizing, to ensure that services are flexible and efficient.

Personal information agents

Many researchers have argued that agents provide such a tool. Pattie Maes from the MIT media lab is perhaps the best-known advocate of this work. She developed a number of prototypical systems that could carry out some of these tasks. I will here describe MAXIMS, an email assistant developed by Maes.

> [MAXIMS] learns to prioritize, delete, forward, sort, and archive mail messages on behalf of a user.
>
> (Maes, 1994a)

MAXIMS works by 'looking over the shoulder' of a user, and learning about how they deal with email. Each time a new event occurs (e.g. email arrives), MAXIMS records the event in the form of

$$\text{situation} \longrightarrow \text{action}$$

pairs. A situation is characterized by the following attributes of an event:

sender of email;

recipients;

subject line;

- keywords in message body and so on.

When a new situation occurs, MAXIMS matches it against previously recorded rules. Using these rules, it then tries to predict what the user will do, and generates a *confidence level:* a real number indicating how confident the agent is in its decision. The confidence level is matched against two preset real number thresholds: a 'tell me' threshold and a 'do it' threshold. If the confidence of the agent in its decision is less than the 'tell me' threshold, then the agent gets feedback from the user on what to do. If the confidence of the agent in its decision is between the 'tell me' and 'do it' thresholds, then the agent makes a suggestion to the user about what to do. Finally, if the agent's confidence is greater than the 'do it' threshold, then the agent takes the initiative, and acts.

Rules can also be hard coded by users (e.g. 'always delete mails from person *X*').MAXIMS has a simple 'personality' (an animated face on the user's GUI), which communicates its 'mental state' to the user: thus the icon smiles when it has made a correct guess, frowns when it has made a mistake, and so on.

The NewT system is a Usenet news filter (Maes, 1994a, pp. 38, 39). A NewT agent is trained by giving it a series of examples, illustrating articles that the user would and would not choose to read. The agent then begins to make suggestions to the user, and is given feedback on its suggestions. NewT agents are not intended to remove human choice, but to represent an extension of the human's wishes: the aim is for the agent to be able to bring to the attention of the user articles of the type that the user has shown a consistent interest in. Similar ideas have been proposed by McGregor, who imagines *prescient agents* - intelligent administrative assistants, that predict our actions, and carry out routine or repetitive administrative procedures on our behalf (McGregor, 1992).

Web agents

Etzioni and Weld (1995) identify the following specific types of Web-based agent they believe are likely to emerge in the near future.

Tour guides. The idea here is to have agents that help to answer the question 'where do I go next' when browsing the Web. Such agents can learn about the user's preferences in the same way that MAXIMS does, and, rather than just providing a single, uniform type of hyperlink, they actually indicate the likely interest of a link.

Indexing agents. Indexing agents will provide an extra layer of abstraction on top of the services provided by search/indexing agents such as Google and *InfoSeek.* The idea is to use the raw information provided by such engines, together

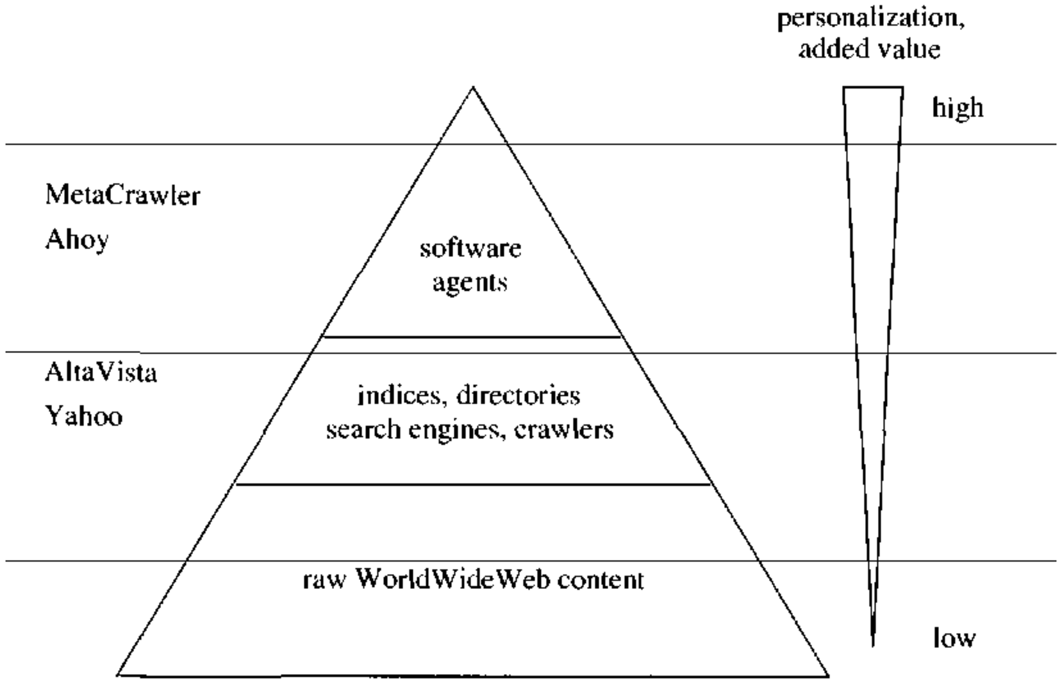

Figure 11.1 The Web information food chain.

with knowledge of the user's goals, preferences, etc., to provide a *personalized* service.

FAQ-finders. The idea here is to direct users to 'Frequently Asked Questions' (FAQs) documents in order to answer specific questions. Since FAQs tend to be knowledge-intensive, structured documents, there is a lot of potential for automated FAQ finders.

Expertise finders. Suppose I want to know about people interested in temporal belief logics. Current Web search tools would simply take the three words 'temporal', 'belief, 'logic', and search on them. This is not ideal: Google has no model of what you *mean* by this search, or what you really *want.* Expertise finders 'try to understand the user's wants and the contents of information services' in order to provide a better information provision service.

Etzioni (1996) put forward a model of information agents that *add value* to the underlying information infrastructure of the Web - the *information food chain* (see Figure 11.1). At the lowest level in the Web information food chain is raw content: the home pages of individuals and companies. The next level up the food chain is the services that 'consume' this raw content. These services include search engines such as Google, Lycos, and Yahoo.

These search engines maintain large databases of Web pages, indexed by content. Apart from the technical difficulties associated with storing such large databases and being able to process and retrieve their contents sufficiently quickly to provide a useful online service, the search engines must also *obtain* and *index* new or changed Web pages on a regular basis. Currently, this is done in one of two ways. The simplest is to have humans search for pages and classify them manually. This has the advantage that the classifications obtained in this way are likely to be meaningful and useful. But it has the very obvious disadvantage that it is not necessarily thorough, and is costly in terms of human resources. The second approach is to use simple software agents, often called *spiders,* to systematically search the Web, following all links, and automatically classifying content. The classification of content is typically done by removing 'noise' words from the page ('the', 'and', etc.), and then attempting to find those words that have the most meaning.

All current search engines, however, suffer from the disadvantage that their coverage is partial. Etzioni (1996) suggested that one way around this is to use a *meta* search engine. This search engine works not by directly maintaining a database of pages, but by querying a number of search engines in parallel. The results from these search engines can then be collated and presented to the user. The meta search engine thus 'feeds' off the other search engines. By allowing the engine to run on the user's machine, it becomes possible to personalize services - to tailor them to the needs of individual users.

Multiagent information retrieval systems

The information resources - Web sites - in the kinds of applications I discussed above are essentially *passive.* They simply deliver specific pages when requested. A common approach is thus to make information resources more 'intelligent' by *wrapping* them with agent capabilities. The structure of such a system is illustrated in Figure 11.2.

In this figure, there are a number of information repositories; these repositories may be Web sites, databases, or any other form of store. Access to these repositories is provided by information agents. These agents, which typically communicate using an agent communication language, are 'experts' about their associated repository. As well as being able to provide access to the repository, they are able to answer 'meta-level' queries about the content ('do you know about *X'?).* The agents will communicate with the repository using whatever native API the repository provides - HTTP, in the case of Web repositories.

To address the issue of finding agents in an open environment like the Internet, *middle agents* or *brokers* are used (Wiederhold, 1992; Kuokka and Harada. 1996). Each agent typically *advertises* its capabilities to some broker. Brokers come in several different types. They may be simply *matchmakers* or *yellow page* agents, which match advertisements to requests for advertised capabilities. Alternatively, they may be *blackboard* agents, which simply collect and make available requests. Or they may do both of these (Decker *et al.,* 1997). Different brokers may be specialized in different areas of expertise. For example, in Figure 11.2, one broker 'knows about' repositories 1,2, and 3; the other knows about 2,3, and 4.

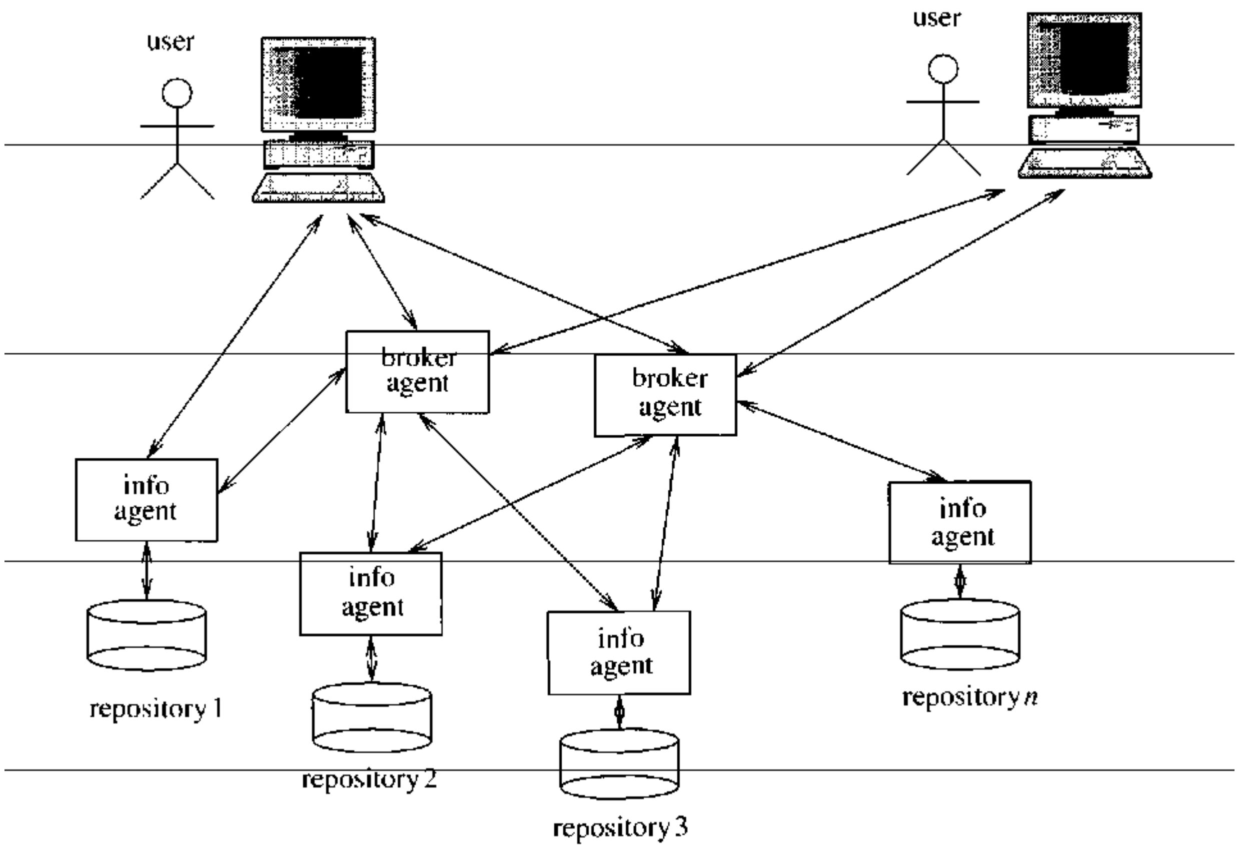

Figure 11.2 Typical architecture of a multiagent information system.

Brokered systems are able to cope more quickly with a rapidly fluctuating agent population. Middle agents allow a system to operate robustly in the face of intermittent communications and agent appearance and disappearance.

The overall behaviour of a system such as that in Figure 11.2 is that a user issues a query to an agent on their local machine. This agent may then contact information agents directly, or it may go to a broker, which is skilled at the appropriate type of request. The broker may then contact a number of information agents, asking first whether they have the correct skills, and then issuing specific queries. This kind of approach has been successfully used in *digital library* applications (Wellman *et al.*, 1996).

11.4 Agents for Electronic Commerce

The boom in interest in the Internet throughout the late 1990s went hand-in-hand with an explosion of interest in electronic commerce (e-commerce) (Ovum, 1994; Guilfoyle *et ai*, 1997). As it currently stands, the Web has a number of features that limit its use as an 'information market'. Many of these stem from the fact that the Web has academic origins, and as such, it was designed for free, open access. The Web was thus not designed to be used for commercial purposes, and a number of issues limit its use for this purpose.

Trust: in an online global marketplace, it is difficult for consumers to know which vendors are reliable/secure and which are not, as they are faced with vendor brands that they have not previously encountered.

Privacy and security: consumers (still) have major worries about the security of their personal information when using e-commerce systems - mechanisms such as secure HTTP (https) go some way to alleviating this problem, but it remains a major issue.

Billing/revenue: no built-in billing mechanisms are provided by the Web - they must be implemented over the basic Web structure; in addition, the Web was not designed with any particular revenue model in mind.

Reliability: the Internet - and hence the Web - is unreliable, in that data and connections are frequently lost, and it thus has unpredictable performance. These limitations may be accepted by academic or home/hobby users, but they represent a very real obstacle in the way of the wider commercial use of the Web.

'First-generation' e-commerce systems (of which amazon.com was perhaps the best known example) allowed a user to browse an online catalogue of products, choose some, and then purchase these selected products using a credit card. However, agents make it possible for *second-generation* e-commerce systems, in which many aspects of a consumer's buying behaviour is automated.

There are many models that attempt to describe consumer buying behaviour. Of these, one of the most popular postulates that consumers tend to engage in the following six steps (Guttman *et al.*,1998, pp. 148, 149).

(1) Need identification. This stage characterizes the consumer becoming aware of some need that is not satisfied.

(2) **Product brokering.** In this stage, a would-be consumer obtains information relating to available products, in order to determine *what* product to buy.

(3) **Merchant brokering.** In this stage, the consumer decides *who* to buy from. This stage will typically involve examining offers from a range of different merchants.

(4) Negotiation. In this stage, the terms of the transaction are agreed between the would-be consumer and the would-be merchant. In some markets (e.g. regular retail markets), the negotiation stage is empty - the terms of agreement are fixed and not negotiable. In other markets (e.g. the used car market), the terms are negotiable.

(5) **Purchase and delivery.** In this stage, the transaction is actually carried out, and the good delivered.

(6) Product service and evaluation. The post-purchase stage involves product service, customer service, etc.

Table 11.1 Current agents for electronic commerce.

	Persona Logic	Firefly	Bargain Finder	Jango	Kasbah	Auction Bot	Tete-a-tete
Need identification	x	x		x			x
Product ~~brokering~~			~~×~~			~~×~~	
Merchant brokering			x	x	x	x	x
Negotiation					x	x	x
Purchase & delivery							
Service & evaluation							

Agents have been widely promoted as being able to automate (or at least partially automate) some of these stages, and hence assist the consumer to reach the best deal possible (Noriega and Sierra, 1999). Table 11.1 (from Guttman *et al.*, 1998) summarizes the extent to which currently developed agents can help in each stage.

Comparison shopping agents

The simplest type of agent for e-commerce is the *comparison shopping* agent. The idea is very similar to the meta search engines discussed above. Suppose you want to purchase the CD ‘Music’ by Madonna. Then a comparison shopping agent will search a number of online shops to find the best deal possible.

Such agents work well when the agent is required to compare goods with respect to a single attribute - typically price. The obvious examples of such situations are 'shrink wrapped' goods such as CDs and books. However, they work less well when there is more than one attribute to consider. An example might be the used-car market, where in addition to considering price, the putative consumer would want to consider the reputation of the merchant, the length and type of any guarantee, and many other attributes.

The Jango system (Doorenbos *et al.*, 1997) is a good example of a first-generation e-commerce agent. The long-term goals of the Jango project were to

- help the user decide what to buy;
- find specifications and reviews of products;
- make recommendations to the user;
- perform comparison shopping for the best buy;
- monitor 'what's new' lists; and
- watch for special offers and discounts.

A key obstacle that the developers of Jango encountered was simply that Web pages are all different. Jango exploited several *regularities* in vendor Web sites in order to make 'intelligent guesses' about their content.

Navigation regularity. Web sites are designed by vendors so that products are easy to find.

Corporate regularity. Web sites are usually designed so that pages have a similar 'look'n'feel';

Vertical separation. Merchants use white space to separate products.

Internally, Jango has two key components:

- a component to *learn vendor descriptions* (i.e. learn about the structure of vendor Web pages); and
- a *comparison shopping* component, capable of comparing products across different vendor sites.

In 'second-generation' agent mediated electronic commerce systems, it is proposed that agents will be able to assist with the fourth stage of the purchasing model set out above: negotiation. The idea is that a would-be consumer delegates the authority to negotiate terms to a software agent. This agent then negotiates with another agent (which may be a software agent or a person) in order to reach an agreement.

There are many obvious hurdles to overcome with respect to this model: The most important of these is *trust.* Consumers will not delegate the authority to negotiate transactions to a software agent unless they trust the agent. In particular, they will need to trust that the agent (i) really understands what they want, and (ii) that the agent is not going to be exploited ('ripped off') by another agent, and end up with a poor agreement.

Comparison shopping agents are particularly interesting because it would seem that, if the user is able to search the entire marketplace for goods at the best price, then the overall effect is to force vendors to push prices as low as possible. Their profit margins are inevitably squeezed, because otherwise potential purchasers would go elsewhere to find their goods.

Auction bots

A highly active related area of work is *auction bots:* agents that can run, and participate in, online auctions for goods. Auction bots make use of the kinds of auction techniques discussed in Chapter 7. A well-known example is the Kasbah system (Chavez and Maes, 1996). The aim of Kasbah was to develop a Web-based system in which users could create agents to buy and sell goods on their behalf. In Kasbah, a user can set three parameters for selling agents:

- desired date to sell the good by;
- desired price to sell at; and
- minimum price to sell at.

Selling agents in Kasbah start by offering the good at the desired price, and as the deadline approaches, this price is systematically reduced to the minimum price fixed by the seller. The user can specify the 'decay' function used to determine the current offer price. Initially, three choices of decay function were offered: linear, quadratic, and cubic decay. The user was always asked to confirm sales, giving them the ultimate right of veto over the behaviour of the agent.

As with selling agents, various parameters could be fixed for buying agents: the date to buy the item by, the desired price, and the maximum price. Again, the user could specify the 'growth' function of price over time.

Agents in Kasbah operate in a *marketplace.* The marketplace manages a number of ongoing auctions. When a buyer or seller enters the marketplace, Kasbah matches up requests for goods against goods on sale, and puts buyers and sellers in touch with one another.

The *Spanish ~~Fishmarket~~* is another example of an online auction system (Rodriguez *et al.*, 1997). Based on a real fishmarket that takes place in the town of Blanes in northern Spain, the FM system provides similar facilities to Kasbah, but is specifically modelled on the auction protocol used in Blanes.

11.5 Agents for Human-Computer Interfaces

Currently, when we interact with a computer via a user interface, we are making use of an interaction paradigm known as *direct manipulation.* Put simply, this means that a computer program (a word processor, for example) will only do something if we explicitly tell it to, for example by clicking on an icon or selecting an item from a menu. When we work with humans on a task, however, the interaction is more two-way: we work with them as peers, each of us carrying out parts of the task and proactively helping each other as problem-solving progresses. In essence, the idea behind interface agents is to make computer systems more like proactive assistants. Thus, the goal is to have computer programs that in certain circumstances could *take the initiative,* rather than wait for the user to spell out exactly what they wanted to do. This leads to the view of computer programs as *cooperating* with a user to achieve a task, rather than acting simply as servants. A program capable of taking the initiative in this way would in effect be operating as a semi-autonomous agent. Such agents are sometimes referred to as *expert assistants* or *interface agents.* Maes (1994b, p. 71) defines interface agents as

> [C]omputer programs that employ artificial intelligence techniques in order to provide assistance to a user dealing with a particular applica-~~tion. . . .~~ The metaphor is that of a *personal assistant* who is *~~collaborat-~~ ing with the user* in the same work environment.

One of the key figures in the development of agent-based interfaces has been Nicholas Negroponte (director of MIT's influential Media Lab). His vision of agents at the interface was set out in his 1995 book *Being Digital* (Negroponte, 1995):

The agent answers the phone, recognizes the callers, disturbs you when appropriate, and may even tell a white lie on your behalf. The same agent is well trained in timing, versed in finding opportune moments, and respectful of idiosyncrasies. ... If you have somebody who knows you well and shares much of your information, that person can act on your behalf very effectively. If your secretary falls ill, it would make no difference if the temping agency could send you Albert Einstein. This issue is not about IQ. It is shared knowledge and the practice of using it in your best interests. ... Like an army commander sending a scout ahead.. .you will dispatch agents to collect information on your behalf. Agents will dispatch agents. The process multiplies. But [this process] started at the interface where you delegated your desires.

[1.6 Agents for Virtual Environments

There is obvious potential for marrying agent technology with that of the cinema, computer games, and virtual reality. The OZ project was initiated to develop:

.. .artistically interesting, highly interactive, simulated worlds.. .to give users the experience of living in (not merely watching) dramatically rich worlds that include moderately competent, emotional agents.

(Bates *et al.*,1992b, p. 1)

In order to construct such simulated worlds, one must first develop *believable agents:* agents that 'provide the illusion of life, thus permitting the audience's suspension of disbelief (Bates, 1994, p. 122). A key component of such agents is *emotion:* agents should not be represented in a computer game or animated film as the flat, featureless characters that appear in current computer games. They need to show emotions; to act and react in a way that resonates in tune with our empathy and understanding of human behaviour. The OZ group investigated various architectures for emotion (Bates *et al.*, 1992a), and have developed at least one prototype implementation of their ideas (Bates, 1994).

•1.7 Agents for Social Simulation

I noted in Chapter 1 that one of the visions behind multiagent systems technology is that of using agents as an experimental tool in the social sciences (Gilbert and Doran, 1994; Gilbert, 1995; Moss and Davidsson, 2001). Put crudely, the idea is that agents can be used to simulate the behaviour of human societies. At its simplest, individual agents can be used to represent individual people; alternatively, individual agents can be used to represent organizations and similar such entities.

Conte and Gilbert (1995, p. 4) suggest that multiagent simulation of social processes can have the following benefits:

- computer simulation allows the observation of properties of a model that may *in principle* be analytically derivable but have not yet been established;
- possible alternatives to a phenomenon observed in nature may be found;
- properties that are difficult/awkward to observe in nature may be studied at leisure in isolation, recorded, and then 'replayed' if necessary;
- 'sociality' can be modelled explicitly - agents can be built that have representations of other agents, and the properties and implications of these representations can be investigated.

Moss and Davidsson (2001, p. 1) succinctly states a case for multiagent simulation:

> [For many systems,] behaviour cannot be predicted by statistical or qualitative analysis. ... Analysing and designing.. .such systems requires a different approach to software engineering and mechanism design.

Moss goes on to give a general critique of approaches that focus on formal analysis at the expense of accepting and attempting to deal with the 'messiness' that is inherent in most multiagent systems of any complexity. It is probably fair to say that his critique might be applied to many of the techniques described in Chapter 7, particularly those that depend upon a 'pure' logical or game-theoretic foundation. There is undoubtedly some strength to these arguments, which echo cautionary comments made by some of the most vocal proponents of game theory (Binmore, 1992, p. 196). In the remainder of this section, I will review one major project in the area of social simulation, and point to some others.

The EOS project

The EOS project, undertaken at the University of Essex in the UK, is a good example of a social simulation system (Doran, 1987; Doran *et ah,* 1992; Doran and Palmer, 1995). The aim of the EOS project was to investigate the causes of the emergence of social complexity in Upper Palaeolithic France. Between 15 000 and 30 000 years ago, at the height of the last ice age, there was a relatively rapid growth in the complexity of societies that existed at this time. The evidence of this social complexity came in the form of Doran and Palmer (1995)

- larger and more abundant archaeological sites;
- increased wealth, density, and stratigraphic complexity of archaeological material in sites;
- more abundant and sophisticated cave art (the well-known caves at Lascaux are an example);

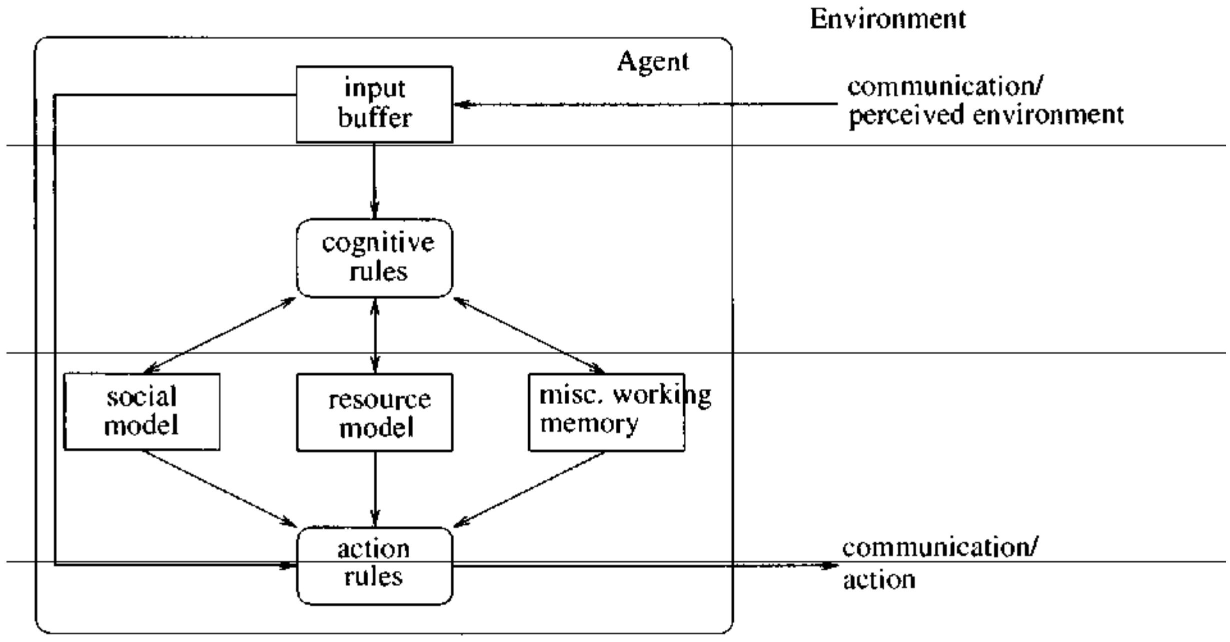

Figure 11.3 Agents in EOS.

- increased stone, bone, and antler technology;
- abundance of 'trade' objects.

A key open question for archaeologists is what exactly *caused* this emergence of complexity. In 1985, the archaeologist Paul Mellars proposed a model that attempted to explain this complexity. The main points of Mellar's model were that the key factors leading to this growth in complexity were an exceptional wealth and diversity of food resources, and a strong, stable, predictable concentration of these resources.

In order to investigate this model, a multiagent experimental platform - the EOS testbed - was developed. This testbed, implemented in the Prolog language (Clocksin and Mellish, 1981), allows agents to be programmed as rule-based systems. The structure of an EOS agent is shown in Figure 11.3.

Each agent in EOS is endowed with a symbolic representation of its environment - its beliefs. Beliefs are composed of beliefs about other agents (the *social model),* beliefs about resources in the environment (the *resource model),* and miscellaneous other beliefs. To update its beliefs, an agent has a set of *cognitive rules,* which map old beliefs to new ones. To decide what action to perform, agents have *action rules:* action rules map beliefs to actions. (Compare with Shoham's AgentO system described in Chapter 3.) Both cognitive rules and action rules are executed in a forward-chaining manner.

Agents in the EOS testbed inhabit a simulated two-dimensional environment, some 10 000 x 10 000 cells in size (cf. the Tileworld described in Chapter 2.) Each agent occupies a single cell, initially allocated at random. Agents have associated with them *skills* (cf. the MACE system described in Chapter 9.) The idea is that an agent will attempt to obtain resources ('food') which are situated in the environment; resources come in different types, and only agents of certain types are able to obtain certain resources. Agents have a number of 'energy stores', and for each of these a 'hunger level'. If the energy store associated with a particular hunger level falls below the value of the hunger level, then the agent will attempt to replenish it by consuming appropriate resources. Agents travel about the EOS world in order to obtain resources, which are scattered about the world. Recall that the Mellars model suggested that the availability of resources at predictable locations and times was a key factor in the growth of the social complexity in the Palaeolithic period. To reflect this, resources (intuitively corresponding to things like a Reindeer herd or a fruit tree) were clustered, and the rules governing the emergence and disappearance of resources reflects this.

The basic form of social structure that emerges in EOS does so because certain resources have associated with them a *skill profile*. This profile defines, for every type of skill or capability that agents may possess, how many agents with this skill are required to obtain the resource. For example, a 'fish' resource might require two 'boat' capabilities; and a 'deer' resource might require a single 'spear' capability.

In each experiment, a user may specify a number of parameters:

- the number of resource locations of each type and their distribution;
- the number of resource instances that each resource location comprises;
- the type of energy that each resource location can supply;
- the quantity of energy an instance of a particular resource can supply;
- the skill profiles for each resource; and
- the 'renewal' period, which elapses between a resource being consumed and being replaced.

To form collaborations in order to obtain resources, agents use a variation of Smith's Contract Net protocol (see Chapter 9): thus, when an agent finds a resource, it can advertise this fact by sending out a broadcast announcement. Agents can then bid to collaborate on obtaining a resource, and the successful bidders then work together to obtain the resource.

A number of social phenomena were observed in running the EOS testbed, for example: ‘overcrowding’, when too many agents attempt to obtain resources in some locale; 'clobbering', when agents accidentally interfere with each other's goals; and semi-permanent groups arising. With respect to the emergence of deep hierarchies of agents, it was determined that the growth of hierarchies depended to a great extent on the perceptual capabilities of the group. If the group is not equipped with adequate perceptual ability, then there is insufficient information to cause a group to form. A second key aspect in the emergence of social structure is the complexity of resources - bow many skills it requires in order to obtain and exploit a resource. If resources are too complex, then groups will not be able to form to exploit them before they expire.

An interesting aspect of the EOS project was that it highlighted the *cognitive* ts of multiagent social simulation. That is, by using EOS, it was possible to see how the beliefs and aspirations of individuals in a society can influence the possible trajectories of this society. One of the arguments in favour of this style of multiagent societal simulation is that this kind of property is very hard to model or understand using analytical techniques such as game or economic theory (cf. the quote from Moss, above).

Policy modelling by multiagent simulation

Another application area for agents in social simulation is that of *policy modelling and development* (Downing *et al,* 2001). Regulatory and other similar bodies put forward policies, which are designed - or at least intended - to have some desired effect. An example might be related to the issue of potential climate change caused by the release of greenhouse gases (cf. Downing *et al,* 2001). A national government, or an international body such as the EU, might desire to reduce the potentially damaging effects of climate change, and put forward a policy designed to limit it. A typical first-cut at such a policy might be to increase fuel taxes, the idea being that this reduces overall fuel consumption, in turn reducing the release of greenhouse gases. But policy makers must generally form their policies in ignorance of what the *actual* effect of their policies will be, and, in particular, the actual effect may be something quite different to that intended. In the greenhouse gas example, the effect of increasing fuel taxes might be to cause consumers to switch to cheaper - dirtier - fuel types, at best causing no overall reduction in the release of greenhouse gases, and potentially even leading to an increase. So, it is proposed, multiagent simulation models might fruitfully be used to gain an understanding of the effect of their nascent policies.

An example of such a system is the Freshwater Integrated Resource Management with Agents (FIRMA) project (Downing *et al.*, 2001). This project is specifically intended to understand the impact of governments exhorting water consumers to exercise care and caution in water use during times of drought (Downing *et al,* 2001, p. 206). (In case you were wondering, yes, droughts *do* happen in the UK!) Downing *et al* (2001) developed a multiagent simulation model in which water consumers were represented by agents, and a 'policy' agent issued exhortations to consume less at times of drought. The authors were able to develop a simulation model that fairly closely resembled the observed behaviour in human societies in similar circumstances; developing this model was an iterative process of model reformulation followed by a review of the observed results of the model with water utilities.

11.8 Agents for *X*

Agents have been proposed for many more application areas than I have the space to discuss here. In this section, I will give a flavour of some of these.

Agents for industrial systems management. Perhaps the largest and probably best-known European multiagent system development project to date was ARCHON (Wittig, 1992; Jennings and Wittig, 1992; Jennings *et al*, 1995). This project developed and deployed multiagent technology in several industrial domains. The most significant of these domains was a power distribution system, which was installed and is currently operational in northern Spain. Agents in ARCHON have two main parts: a *domain* component, which realizes the domain-specific functionality of the agent; and a *wrapper* component, which provides the agent functionality, enabling the system to plan its actions, and to represent and communicate with other agents. The ARCHON technology has subsequently been deployed in several other domains, including particle accelerator control. (ARCHON was the platform through which Jennings's joint intention model of cooperation (Jennings, 1995), discussed in Chapter 9, was developed.)

Agents for Spacecraft Control. It is difficult to imagine a domain with harder real-time constraints than that of in-flight diagnosis of faults on a spacecraft. Yet one of the earliest applications of the PRS architecture was precisely this (Georgeff and Lansky, 1987). In brief, the procedures that an astronaut would use to diagnose faults in the Space Shuttle's reaction control systems were directly coded as PRS plans, and the PRS architecture was used to interpret these plans, and provide real-time advice to astronauts in the event of failure or malfunction in this system.

Agents for Air-Traffic Control. Air-traffic control systems are among the oldest application areas in multiagent systems (Steeb *et al*, 1988; Findler and Lo, 1986). A recent example is OASIS (Optimal Aircraft Sequencing using intelligent Scheduling), a system that is currently undergoing field trials at Sydney airport in Australia (Ljunberg and Lucas. 1992). The specific aim of OASIS is to assist an air-traffic controller in managing the flow of aircraft at an airport: it offers estimates of aircraft arrival times, monitors aircraft progress against previously derived estimates, informs the air-traffic controller of any errors, and perhaps most importantly finds the optimal sequence in which to land aircraft. OASIS contains two types of agents: *global* agents, which perform generic domain functions (for example, there is a 'sequencer agent', which is responsible for arranging aircraft into a least-cost sequence); and *aircraft agents,* one for each aircraft in the system airspace. The OASIS system was implemented using the PRS agent architecture.

Notes and Further Reading

Jennings and Wooldridge (1998a) is a collection of papers on applications of agent systems. Parunak (1999) gives a more recent overview of industrial applications.

Hayzelden and Bigham (1999) is a collection of articles loosely based around the theme of agents for computer network applications; Klusch (1999) is a similar collection centred around the topic of information agents.

Van Dyke Parunak (1987) describes the use of the Contract Net protocol (Chapter 8) for manufacturing control in the YAMS {*Yet* Another Manufacturing System). Mori *et al.* have used a multiagent approach to controlling a steel coil processing plant (Mori *et al,* 1988), and Wooldridge *et al* have described how the process of determining an optimal production sequence for some factory can naturally be viewed as a problem of negotiation between the various production cells within the factory (Wooldridge *et al,* 1996).

A number of studies have been made of information agents, including a theoretical study of how agents are able to incorporate information from different sources (Levy *et al,* 1994; Gruber, 1991), as well as a prototype system called IRA (information retrieval agent) that is able to search for loosely specified articles from a range of document repositories (Voorhees, 1994). Another important system in this area was Carnot (Huhns *et al,* 1992), which allows preexisting and heterogeneous database systems to work together to answer queries that are outside the scope of any of the individual databases.

There is much related work being done by the computer supported cooperative work (CSCW) community. CSCW is informally defined by Baecker to be 'computer assisted coordinated activity such as problem solving and communication carried out by a group of collaborating individuals' (Baecker, 1993, p. 1). The primary emphasis of CSCW is on the development of (hardware and) software tools to support collaborative human work - the term *groupware* has been coined to describe such tools. Various authors have proposed the use of agent technology in groupware. For example, in his *participant systems* proposal, Chang suggests systems in which humans collaborate with not only other humans, but also with artificial agents (Chang, 1987). We refer the interested reader to the collection of papers edited by Baecker (1993) and the article by Greif (1994) for more details on CSCW.

Noriega and Sierra (1999) is a collection of paper on agent-mediated electronic commerce. Kephart and Greenwald (1999) investigates the dynamics of systems in which buyers and sellers are agents.

Gilbert and Doran (1994), Gilbert and Conte (1995) and Moss and Davidsson (2001) are collections of papers on the subject of simulating societies by means of multiagent systems. Davidsson (2001) discusses the relationship between multiagent simulation and other types of simulation (e.g. object-oriented simulation and discrete event models).

Class reading: Parunak (1999). This paper gives an overview of the use of agents in industry from one of the pioneers of agent applications.