IEE Review, January 1996, pp 17-20

Software Agents

Nick Jennings
Dept. of Electronic Engineering,
Queen Mary & Westfield College,
Mile End Road, London E1 4NS.
N.R.Jennings@qmw.ac.uk

Michael Wooldridge
Dept. of Computing,
Manchester Metropolitan University,
Chester Street, Manchester M1 5GD.
M.Wooldridge@mmu.ac.uk

1. WHAT ARE SOFTWARE AGENTS?

Software agents are probably the fastest growing area of Information Technology (IT). They are being used, and touted, for applications as diverse as personalised information management, electronic commerce, interface design, computer games, and management of complex commercial and industrial processes. Despite this proliferation, there is, as yet, no commonly agreed upon definition of exactly what an agent is — Smith *et al.* (1994) define it as "a persistent software entity dedicated to a specific purpose"; Selker (1994) takes agents to be "computer programs that simulate a human relationship by doing something that another person could do for you"; and Janca (1995) defines an agent as "a software entity to which tasks can be delegated". To capture this variety, a relatively loose notion of an agent as a self-contained program capable of controlling its own decision making and acting, based on its perception of its environment, in pursuit of one or more objectives will be used here.

Within the extant applications, three distinct classes of agent can be identified. At the simplest level, there are "*gopher*" agents, which execute straightforward tasks based on pre-specified rules and assumptions (eg inform me when the share price deviates by 10% from its mean position or tell me when I need to reorder stock items). The next level of sophistication involves "*service performing*" agents, which execute a well defined task at the request of a user (eg find me the cheapest flight to Paris or arrange a meeting with the managing director some day next week). Finally, there are "*predictive*" agents, which volunteer information or services to a user, without being explicitly asked, whenever it is deemed appropriate (eg an agent may monitor newsgroups on the INTERNET and return discussions that it believes to be of interest to the user or a holiday agent may inform its user that a travel firm is offering large discounts on holidays to South Africa knowing that the user is interested in safaris). Common to all these classes are the following key hallmarks of agenthood (Wooldridge and Jennings, 1995):

- *Autonomy*: agents should be able to perform the majority of their problem solving tasks without the direct intervention of humans or other agents, and they should have a degree of control over their own actions and their own internal state.

- *Social ability*: agents should be able to interact, when they deem appropriate, with other software agents and humans in order to complete their own problem solving and to help others with their activities where appropriate.

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

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

When taken together, these attributes mark software agents as a fundamentally new paradigm — markedly different from related IT disciplines such as object-oriented systems, artificial intelligence, and distributed computing. There are two key distinguishing characteristics. Firstly, relatively high-level tasks can be *delegated* to agents who will *autonomously* carry them out. Secondly, agents are *situated* in an environment which can dynamically affect their problem solving behaviour and strategy. When a new task is delegated, the agent has to determine precisely what the objective is, evaluate how this objective can be reached in an effective manner, and perform the necessary actions. Whilst this is going on, the agent must keep track of what is happening in its environment to ensure that the objective is still appropriate, that the plan of action is still valid, and that no new, and more important, opportunities have arisen. As well as these internal processes, software agents often need to interact with other entities (either humans or software agents) in order to accomplish their objectives. Such interchanges may range from simple requests (eg tell me how much a particular flight to Paris costs) to complex negotiations (eg arguing over which date to arrange a meeting).

2. SOME AGENT APPLICATIONS

This section briefly examines three of the most promising areas in which service performing and predictive software agents are being used. These applications were chosen to give a flavour for the usefulness and power of the technology and should not be considered as an exhaustive list. Other areas which have been identified include: messaging software, development tools, information management and retrieval, user interface software, process control, workflow management, and network management (Guilfoyle and Warner, 1994; Janca, 1995).

2.1 Personal Information Management

The rate of growth of network technology, and in particular the number of commercial and academic organisations that are beginning to make routine use of the INTERNET, has surprised even computer network professionals. Nobody could have predicted in 1991 that within the space of four years, up to a third of all INTERNET traffic would derive from a new network application that did not even exist at that time. And yet that is exactly the situation with the World Wide Web (WWW). This explosion of interest in the INTERNET and its associated technologies has brought with it astonishing potential. As academics, this potential is evident to us in the way that research has changed. It is common for new papers and results to be disseminated electronically, either via mailing lists, or more usually via the WWW. Whereas just two decades ago, it could take years for research results to filter down through academic journals to the typical researcher, new results (and even software) can now be distributed within days or even hours. But the richness and diversity of information sources has brought with it problems. Firstly, it is sometimes difficult to *find* the appropriate information (witness the problem of being "lost in hyperspace" when using the WWW). More generally, however, it is a problem simply to sift through the mass of information available to actually find what you *need* (witness the problem of reading a popular electronic newsgroup).

What would be ideal to help us take advantage of the potential of the INTERNET (and network technology in general) is a trained assistant: someone who constantly searches and sifts the available sources of information on our behalf; someone who knows what we are interested in, and what we are not interested in; someone who will chase up interesting leads, or follow WWW links to new sites, to find out if they contain anything of interest; someone who can methodically search for a poorly specified article in a number of WWW sites without getting bored or distracted by links to the "Playboy/girl" home page; someone who could scan our email for us, sort it into an order that we consider important, and even junk email that is of no interest. Of course, in the real world, such assistants do not exist. Even if they did, the cost of hiring them would almost certainly make them impractical. This is where the idea of software agents as personal information managers comes in.

In pioneering work, done primarily by Pattie Maes' group at MIT (Maes, 1994), prototypes of exactly such applications have been developed. For example, Maes' group have described a prototype electronic news scanner called NEWT. This scanner is trained by a user to pick out certain articles of news from various newsgroups, so that after a while, NEWT will be able to consistently suggest articles that the user is interested in. The idea is not that NEWT tells the user what to read, but that it acts as an *extension* of the user, and acts in accordance with the user's wishes and intentions.

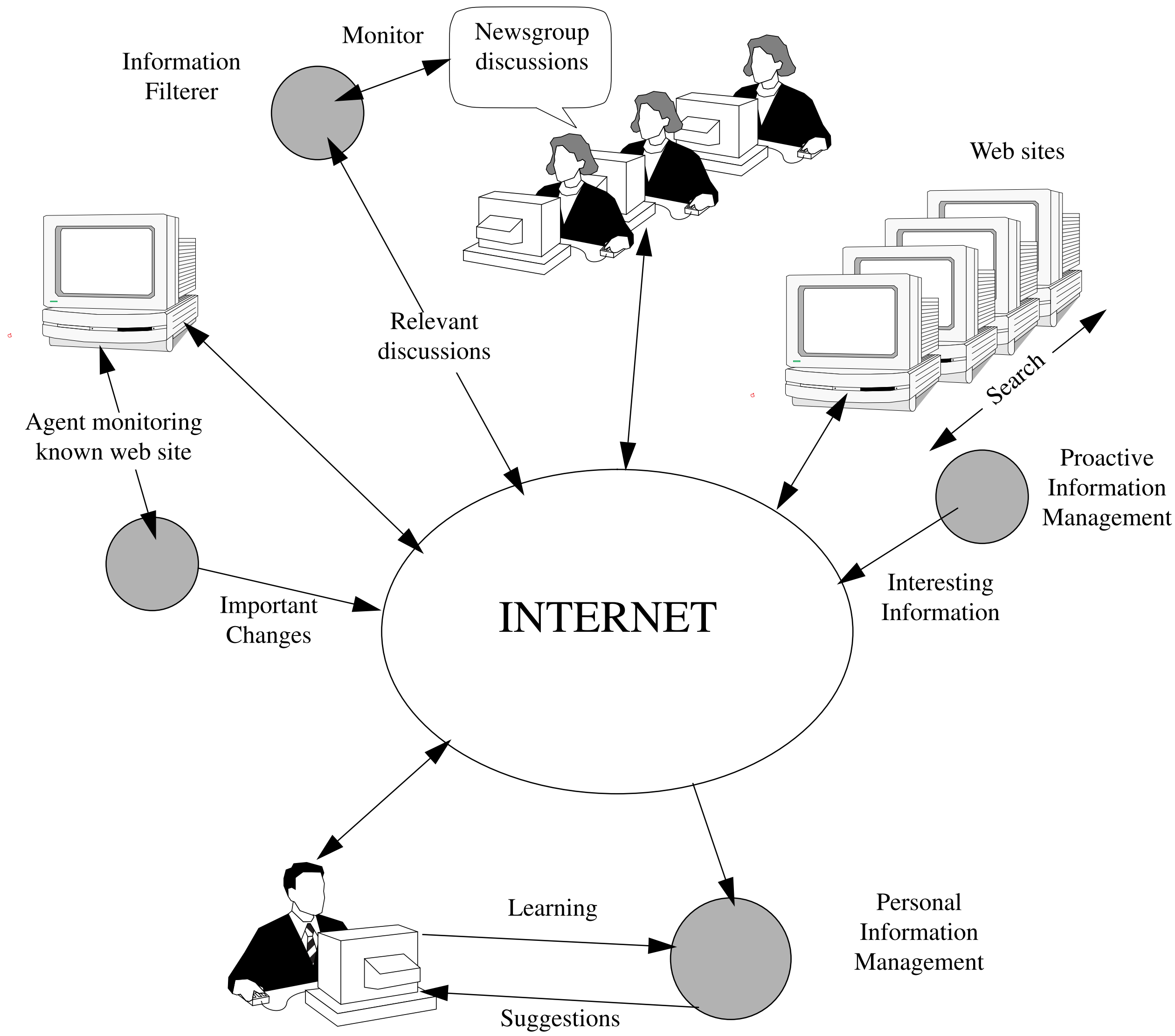

Figure 1: Personal Information Management

NEWT is an example of an *information filter* (see figure 1). Essentially, it sieves information sources to allow through to the user only information that the user would choose to see. More exciting — but also much harder to achieve — is the idea of a *proactive information management agent*. By this, we mean an agent that does not just filter information, but actually goes out and fetches relevant information on the user's behalf. To see how one might work, consider a user that more-or-less regularly checks a variety of WWW sites, to see if they have anything new. (The WWW sites might be academic, in which case they may contain research papers; alternatively, they may be commercial, containing, say, airline schedules and details of special offers.) Every time the user starts using the WWW, the agent watches what links are followed, and what documents are retrieved. The agent tries to learn about the user's browsing behaviour. Eventually, the agent notices that on a particular home page, the user always downloads new papers by a certain author. Once this is established, the agent can begin to do this itself, on behalf of the user. Upon logging in, the user is presented with a list of new, relevant research papers. Thus everything the user does becomes a lesson for the agent. Clearly, this kind of agent poses some fairly big problems for the machine learning community. However, one of the key lessons learned in artificial intelligence over the years is that a little intelligence goes a long way: even an agent that kept the user informed of WWW pages that had been changed or updated would be potentially very useful, by drawing the user's attention to new information.

The goal of information management agents is quite simple: to increase human productivity. The intent is to do this by firstly allowing the user to *focus* on the *right* information, and secondly by giving the user access to information that would otherwise be too costly to obtain. We might draw an analogy with mechanisation in the factory and the office. Clearly, hardware robots have had an enormous impact on the productivity of manufacturing industries (such as car making). However, computerisation has not had a comparable impact on the office: while office work has undoubtedly been made more efficient, by the use of tools such as word processors and spreadsheets, there is still enormous scope for improvement. The kind of information management software agents that we described above may ultimately have the same impact on the life of the office that hardware robots had on the factory floor.

2.2 Electronic Commerce

As well as providing information repositories, the INTERNET is increasingly being used to buy and sell goods and services. Currently, for example, it is possible to order things like pizzas and compact discs electronically although it is inevitable that increasingly expensive goods and services will also come on line in this way. In such environments, software agents are an integral part of the overall system and method of approach. There are *user agents* acting on behalf of the consumer and *business agents* representing the suppliers. Although currently beyond the state of the art, it will not be long before users will be able to delegate to their software agent the task of getting cinema tickets to see one of the latest releases on Saturday night and then booking a meal in a nearby restaurant afterwards. Given this objective and knowledge of its user's preferences (eg vegetarian, no spicy food, comedies rather than action films, etc.), the proxy will enter the appropriate electronic marketplace where it will encounter the business agents (see figure 2). The proxy will first query the cinema agents to determine the most appropriate film and the most convenient showing time. Having made this decision, the proxy will then progress onto the restaurant agents to see which one best fits with the user's preferences and constraints. If no appropriate restaurant can be found, the proxy will return to the cinema agent pool and reassesses its choice of film or show time and then repeat the process. Eventually, if successful, the proxy will report back to the user with the cinema and restaurant booking information. Again in this application, the role of the software proxy is to make informed decisions on behalf of the user. It saves the user from having to go through a series of relatively mundane, but potentially time consuming, activities and thus frees up their time for more rewarding endeavours.

In this scenario, the proxies are examples of service performing agents. They exhibit all facets of agenthood described earlier — they are delegated a high-level task which they have to complete on their own (autonomy), they need to interact and negotiate with other service providing agents (social ability), if the preferred restaurant is fully booked the proxy should try another one (responsiveness), and the proxy might voluntarily check with the railway company agent to find out the time of the last train home from the station nearest the restaurant (proactiveness). In addition to these core attributes, agents in this type of application might well be *mobile*. That is, they physically move from executing on the user's machine onto the machine(s) where the electronic marketplace is operating. Such movement decreases the communication costs because network traffic is reduced (all queries are local); however it increases the security measures which must be incorporated into the system (since mobile agents bear most of the hallmarks of computer viruses!)

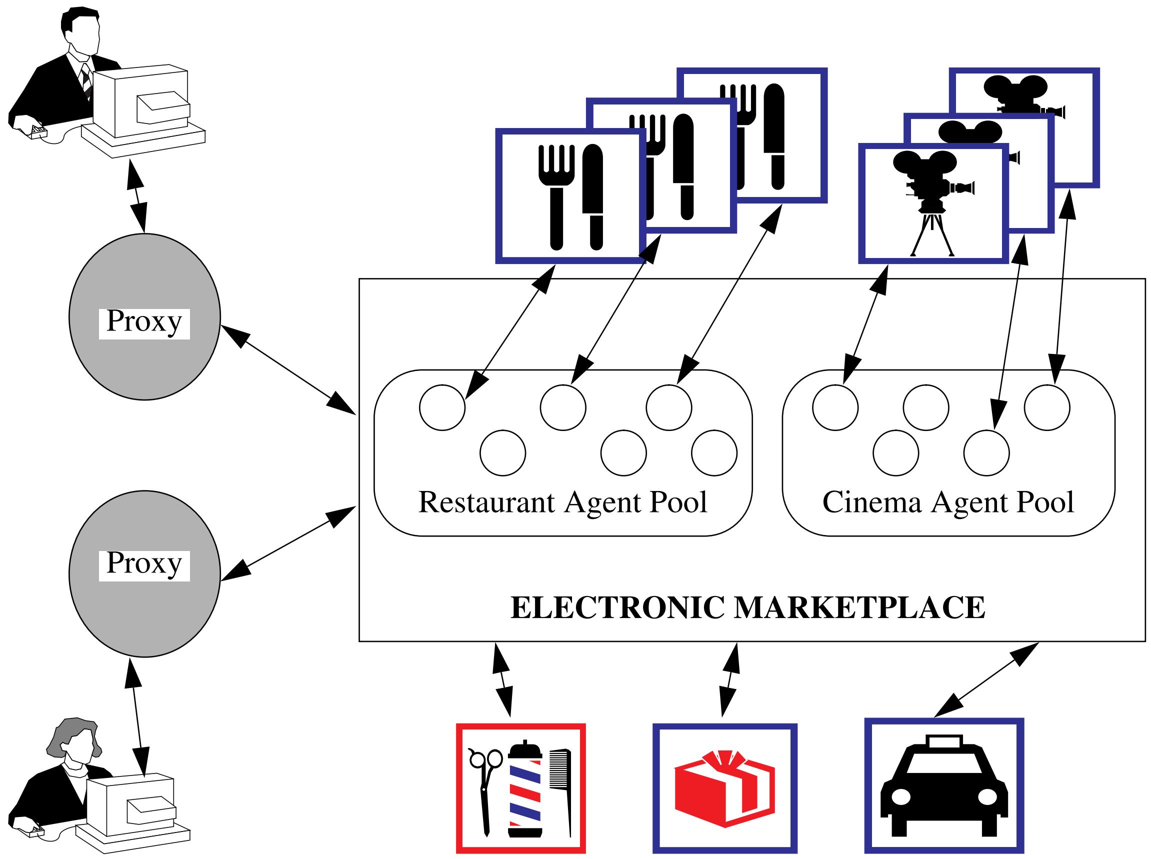

Figure 2: The Electronic Marketplace

2.3 Business Process Management

Software agents do not only empower individuals, they are also being used to ensure entire organisations operate more effectively. The reason for their deployment is that many business processes — such as designing a new aircraft, building a shopping centre, or tendering for large contracts — are becoming increasingly complex. They typically involve a number of individuals, located in a number of different departments, who need to work together in order to get a task done. Thus, for example, when tendering for a large contract a company needs to bring in its legal department (to make sure the bid is legal), its technical department (to ensure the bid specification is met), its marketing department (to ensure the bid is presented in the best possible light), and so on. Traditionally, the management of such processes has been done manually — however this leads to activities being forgotten and the wrong information being sent to the wrong people at the wrong time. It is also very expensive and resource intensive. Recently, organisations have started to employ limited computer support, such as workflow tools, to automate some of these exchanges. However such tools are typically very rigid and unresponsive to changes in circumstances or unexpected events.

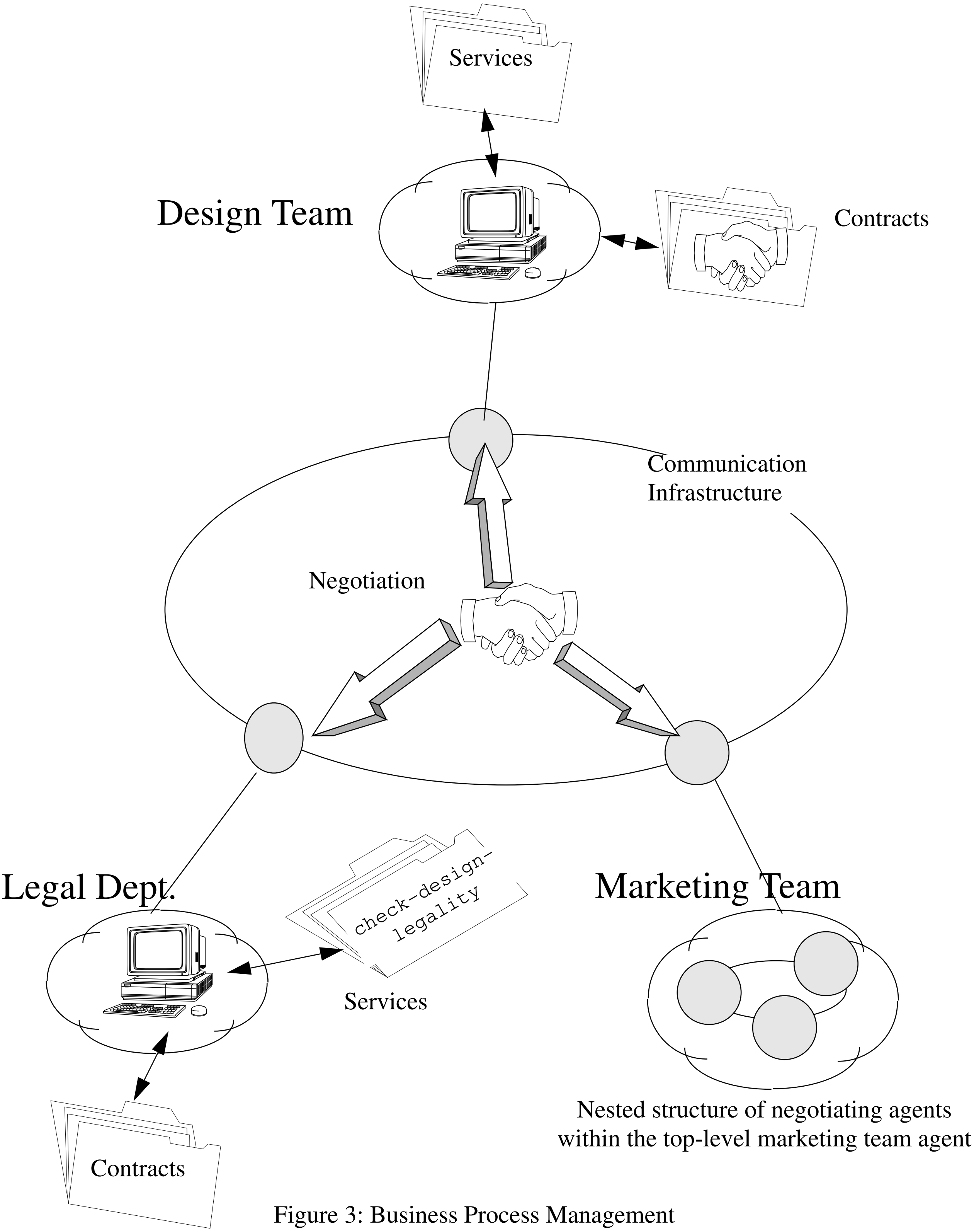

Figure 3: Business Process Management

To better manage the flow of information and the scheduling of activities, a number of projects, and in particular ADEPT (Alty *et al.*, 1994), have adopted a negotiating agents approach. At the topmost level each department is represented by a software agent (see figure 3). Agents are autonomous problem solving entities capable of performing *services* — thus the technical department can perform the service `produce-design` and the legal department can perform the service `check-design-legality`. The internal structure and organisation of each department can be represented in a recursive manner as another set of negotiating agents.

The particular business process is started by the user delegating a high-level task (eg producing a tender) to the agent system. In this case, as part of the delegation, the technical department agent will realise that it has to execute the `produce-design` service. However it knows that before this task can be completed, the design must be checked by the legal department. Thus before the check is actually needed the technical agent will initiate a negotiation with the legal agent to fix a time at which the requisite service can be invoked. Assuming the agents are able to reach an agreement, a binding contract will come into existence. At the appropriate time, and under the conditions specified in the contract, the technical agent will make a request to the legal agent to invoke the desired service. Once the legal agent completes the service it will duly report the necessary results back to the technical agent so that appropriate actions can be taken.

Representing business functions in this manner allows the autonomy of the departments to be preserved. It is also a natural way of representing the distribution of data, control, and resources which occur within all large business activities. Negotiating agents offer a flexible and responsive approach to the problem, agents make agreements in a “just in time” fashion and thus resources can be used more effectively.

3. PROBLEMS AND PROMISES

In this brief article, we have tried to convey some of the key concepts and applications of the vibrant field of software agents — for a more in depth analysis refer to Wooldridge and Jennings (1995). Software agents are currently being used in hundreds of applications, both to solve new types of problems (such as personal information management and electronic commerce) and more traditional problems (such as business process management and network management). Moreover, two recent market surveys have concluded that agents will be the most important computing paradigm in the next 10 years — Janca (1995) claims that “by the year 2000 every significant application will have some form of agent functionality”, while Guilfoyle and Warner (1994) estimate that the total value of agent related markets in the US and Europe could be worth 1.2 billion pounds by the year 2000. Such significance is attached to intelligent agents because the metaphor of software as a sophisticated assistant capable of autonomously solving the user’s goals is intuitively appealing, computationally powerful, and makes software accessible to non-computer specialists. Another important aspect of this metaphor is that the agents can be personalised to reflect the user’s needs, preferences, and constraints.

Amongst all this optimism, it is important that the claims made about software agents are realistic and responsible — they are presently being hyped to a dangerously high level. Software agents will **not** be all pervasive. They will not magically solve all the difficult problems which exist in the current generation of advanced information processing systems — eg planning in uncertain environments, perceiving and acting in a timely fashion in response to environmental changes, and inferring a user’s preferences based on their behaviour. Moreover, by their very nature software agents create a new set of problems which must be tackled (Norman, 1994). Because they are autonomous, users may be wary in trusting them to act on their behalf. For example, even a relatively benign email filtering agent may delete an exceedingly important message which causes considerable loss to its user. However this becomes even worse when agents make financial commitments on behalf of their user. Thus it is vital that appropriate safeguards are built into the software so that agents do not overstep their jurisdiction.

REFERENCES

J. L. Alty, D. Griffiths, N. R. Jennings, E. H. Mamdani, A. Struthers, and M. E. Wiegand (1994) "ADEPT - Advanced Decision Environment for Process Tasks: Overview & Architecture" Proc. BCS Expert Systems 94 Conference (Applications Track), Cambridge, UK, 359-371.

C. Guilfoyle and E. Warner (1994) "Intelligent Agents: The New Revolution in Software" Ovum Report.

P. C. Janca (1995) "Pragmatic Application of Information Agents: BIS Strategic Decisions.

P. Maes, (1994) "Agents that reduce work and information overload" Comms. of the ACM 37 (7) pp 30-40.

D. A. Norman, (1994) "How might people interact with agents" Comms. of the ACM 37 (7) pp 68-71.

T. Selker (1994) "A Teaching Agent that learns" Communications of the ACM 37 (7) pp 92-99.

D. C. Smith, A. Cypher and J. Spohrer (1994) "Programming Agents without a programming language" Communications of the ACM 37 (7) pp 55-67.

M. J. Wooldridge, and N. R. Jennings, (1995) "Intelligent Agents: Theory and Practice" The Knowledge Engineering Review 10 (2).