Applying Agent Technology

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

Michael Wooldridge
Department of Computing
Manchester Metropolitan University
Chester Street, Manchester M1 5GD
United Kingdom
M.Wooldridge@doc.mmu.ac.uk

1. Intelligent Agents and Multi-Agent Systems

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

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

- *Autonomy*: agents should be able to perform the majority of their problem solving tasks without the direct intervention of humans or other agents, and they should have a degree of control over their own actions and their own internal state (Castelfranchi, 1995).
- *Social ability*: agents should be able to interact, when they deem appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities. This requires that agents have, as a minimum, a means by which they can communicate their requirements to others and an internal mechanism for deciding when social interactions are appropriate (both in terms of generating appropriate requests and judging incoming requests).
- *Responsiveness*: agents should perceive their environment (which may be the physical world, a user, a collection of agents, the INTERNET, etc.) and respond in a timely fashion to changes which occur in it.
- *Proactiveness*: agents should not simply act in response to their environment, they

should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where it is appropriate.

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

Having defined some key characteristics of an agent, the next step is to identify why, in various quarters, agent based computing has been hailed as “the next significant breakthrough in software development” (Sargent, 1992) and “the new revolution in software” (Ovum, 1994). Indeed according to one UK market researcher, the worldwide market for agent software will grow from an annual value of $37 million in 1994 to $2.6 billion by the year 2000 (see Figure 1 for a more detailed breakdown of these figures).

<figure 1>

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

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

- Openness

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

- Complexity

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

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

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

- Distribution of data, control, expertise or resources

  When the domain involves a number of distinct problem solving entities (or data sources) which are physically or logically distributed (in terms of their data, control, expertise or resources) and which need to interact with one another (or be combined) in order to solve their problems (or one common problem), then agents can often provide an effective solution. For example, in a distributed health care setting general practitioners, hospital specialists, nurses, and home care organisations have to work together to provide the appropriate care to a sick patient (Huang *et al*., 1995). In this scenario, which is well suited to an agent based solution, there is a distribution of: data (the general practitioner has his own data about the patient which is very different from that of the hospital nurse even though it concerns the same person), control (each individual is responsible for performing a different set of tasks), expertise (the specialist's knowledge is very different from that of either the general practitioner or the nurse), and resources (a specialist is responsible for the beds which his patients need, a general practitioner for paying for the hospital services, etc.).

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

- Natural Metaphor

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

- Legacy Systems

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

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

2. Agent Applications: Promises and Problems

The number of agent based applications being developed and deployed in real world settings is rapidly increasing. Exemplar systems from the field of information management, one of the fastest growing application areas, include:

- The White House has a system which uses intelligent agents to automatically retrieve information in response to the hundreds of requests that it receives via INTERNET everyday (cited in Houlder, 1994). Agents are used to match keywords in the email received with relevant mailing lists of which they are aware.

- The Commander Exception Monitor (produced by Andersen Consulting and Comshare) uses agents to filter information (cited in Houlder, 1994). This product has been used by Hertz, the car rental company, to analyse pricing structures in the car rental business. By excluding trivial changes, the system reduces the equivalent of 28,000 spreadsheets of information about prices, locations, sizes and type of car into something that can easily be monitored by its pricing executives.

- AT&T's PersonaLink (and the as yet unnamed IBM Intelligent Communication Services system) are both based on agent technology (Reinhardt, 1994). The former uses agents to filter messages and search for information on a network, while the latter is an umbrella for smart message routing.

- Legent's AgentWorks uses agents to manage remote resources from a central location, letting network administrators support large, distributed user populations (IEEE Expert, 1994).

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

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

The papers in this special issue clearly illustrate both these positive and negative aspects of an agent based approach. They provide a representative sample of the types of applications and the types of agents which are currently being developed and deployed. The agents vary from complex decision support systems to entities which can only decide about the movement of a single robot joint; they employ problem solving mechanisms which vary from purely deliberative (explicit models of the world and other agents which are reasoned about using declarative knowledge) to purely reactive (no explicit models of the environment and behaviour determined by matching observed conditions with pre-defined responses) via hybrid systems; and, finally, they vary in terms of their level of generality from systems in which the structures, mechanisms and knowledge can be re-used to systems which are hand-crafted and entirely specific to a particular application. The applications cover both situations in which all the agents are working together towards a single global goal and situations in which each individual agent has its own individual goals (and the community, therefore, may have a number of conflicting goals). The exemplar problem domains are open, complex, and distributed, some involve legacy systems and some use agents because they are the most natural means of modelling the system.

The first two papers involve relatively complex deliberative agents which are not operating in a hard real time environment (although in both cases the time taken to act is of some importance). Mason describes an agent based system whose ultimate aim is to automate the interpretation of data emanating from a global network of seismic situations in the context of nuclear test ban treaty verification. The problem involves providing support for a community of seismologists: their agents cooperate by exchanging partial results that act as clues about what assumptions to make and where the computationally expensive signal processing algorithms could best be applied. Huang *et al.* describe a system that provides decision support for distributed health care management in real world settings. The agents make sure that clinical tasks are carried out as agreed, and that information flows appropriately (if not the agents will automatically do something about it). By having a sophisticated means of dealing with incomplete and conflicting viewpoints, the agents can also help with decision support tasks such as deciding which course of treatment a particular patient should follow and what drugs a patient should be prescribed.

The papers of Ferguson and Fischer *et al.* describe hybrid agent architectures. Both architectures are layered and these layers vary from entirely reactive to entirely deliberative. The reactive components are needed to deal with the real time requirements placed on them by their respective domains of road navigation and transportation management. In both cases, however, deliberative capabilities are also needed to ensure the agents exhibit appropriate goal-oriented behaviour. Ferguson's design was shaped by the desire to develop goal directed agents which could respond to unanticipated events in the environment, and could coordinate with other agents. His architecture involves a number of concurrently operating, latency bounded, task achieving layers, and aims to strike a balance between planning far ahead, (which will invariably necessitate replanning in complex and open environments), and leaving decisions as late as possible (which may lead to non-strategic decision making and hence failure to achieve the desired goals). Goal directed behaviour is achieved by making predictions based on the beliefs, desires and intentions of other agents - this activity is made more difficult because the openness of the domain means that an agent's initial knowledge of its world and of other agents is severely limited. Fischer *et al.*'s hybrid architecture bears many similarities to that of Ferguson, although it additionally includes features for handling truly joint actions. It is demonstrated on the application of cooperative order scheduling within a society of shipping companies. Their on-line multi-agent system, based on anytime algorithms, represents an advance over traditional operational research and numerical computing approaches in that it can cope with situations in which no complete specification of tasks is available *a priori*. This improvement is made possible by devolving the problem solving to the autonomous agents - the solution to the global problem emerges out of the local decision making and problem solving strategies - meaning more specialised and bounded reasoning and action can be undertaken by the distributed entities.

The final two papers describe the development of reactive agent systems for the applications of robot motion planning and game playing. Overgaard *et al.*'s approach to robot motion planning and collision avoidance eschews traditional centralised techniques and instead implements the links and joints of a single robot as autonomous agents. Although the agents are purely reactive and selfishly attempt to optimise their own utility, the behaviour of the overall system gives the impression of one which is intelligently controlled. Their method has been used in an industrial application (at Odense Steel Shipyard) to control welding robots installations for ship building with up to 11 degrees of freedom. Wavish and Graham describe a way of organising systems of interacting agents and a corresponding agent architecture and implementation technology based on the situated action approach. They also report on their experiences of applying this technology to the production of an agent based game embedded in a published CD-i title. Their ultimate objective is to make games more interactive and more dependent on what the user wants to do. This means the characters (naturally modelled as agents) should display purposeful and emotional behaviour and appear to react appropriately in the given situation. In addition to getting the agents to operate in real-time, this approach had to contend with the limited processing power available on current games platforms.

3. Future Challenges and Open Problems

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

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

  Surprising little work has been undertaken on methodological aspects of agent based systems, and yet if this technology is to be a commercial success then designers must have a structured way of developing well-engineered agents and agent systems. This work needs to identify how robust and flexible individual agents can best be designed and how these well designed components can then be combined to give an overall system which is similarly robust and flexible. The problem of providing a system level description is made more difficult by the fact that many of its properties can only be observed at run-time. Note that some attempts have been made to investigate formal methods for agent-based systems (Wooldridge, 1992; Wooldridge, 1995). This preliminary work has been successful in identifying the key issues in the formal specification and verification of agent based systems, and a number of simple demonstrator applications and multi-agent languages have been specified. However, as is the case in mainstream computer science, the complexity of such methods means that their use is not likely to become commonplace in the immediate future.

- The development of benchmarks for evaluating design options

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

- The development of reusable tools

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

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

- Heterogeneity

  Agent based research is only just beginning to grapple with problems associated with the inevitable heterogeneity of its problem solving components. The basic problem is how agents with different domains of discourse, employing different knowledge representation schemes, different problem solving paradigms, and with different assumptions about their world and each other, can be made to interact in an effective and scalable manner.

- Reasoning with uncertain, incomplete and contradictory information

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

- Real time operation

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

- Adaptability

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

REFERENCES

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

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

C. Castelfranchi. 1995. Guarantees for autonomy in cognitive agent architecture. In Intelligent Agents - Theories, Architectures, and Languages, (eds. M. Wooldridge and N. R. Jennings). Springer-Verlag Lecture Notes in AI Volume 890. pp 56-70.

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

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

I. A. Ferguson. 1995. On the role of BDI Modelling for integrated control and coordinated behaviour in autonomous agents. In this volume.

K. Fischer, J. P. Müller and M. Pischel. 1995. Cooperative transportation scheduling: an application domain for DAI. In this volume.

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

V. Houlder. 1994. Special agents. Financial Times Technology Section, 15th August, p 12.

J. Huang, N. R. Jennings, and J. Fox. 1995. An agent based approach to health care management. In this volume.

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

IEEE Expert. 1994. Intelligent agents manage distributed systems. IEEE Expert, October pp 69-70.

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

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

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

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

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

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

C. L. Mason. 1995. Cooperative interpretation of seismic data for nuclear test ban treaty verification: a DAI approach. In this volume.

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

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

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

L. Overgaard, H. G. Petersen and J. W. Perram. 1995. Reactive motion planning: a multi-agent approach. In this volume.

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

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

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

A. Reinhardt. 1994. The network with smarts. Byte, October, pp 51-64.

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

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

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

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

P. Wavish and M. Graham. 1995. A situated action approach to implementing characters in computer games. In this volume.

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

M. Wooldridge. 1992. The Logical Modelling of Computational Multi-Agent Systems. PhD thesis, UMIST, Manchester, October 1992.

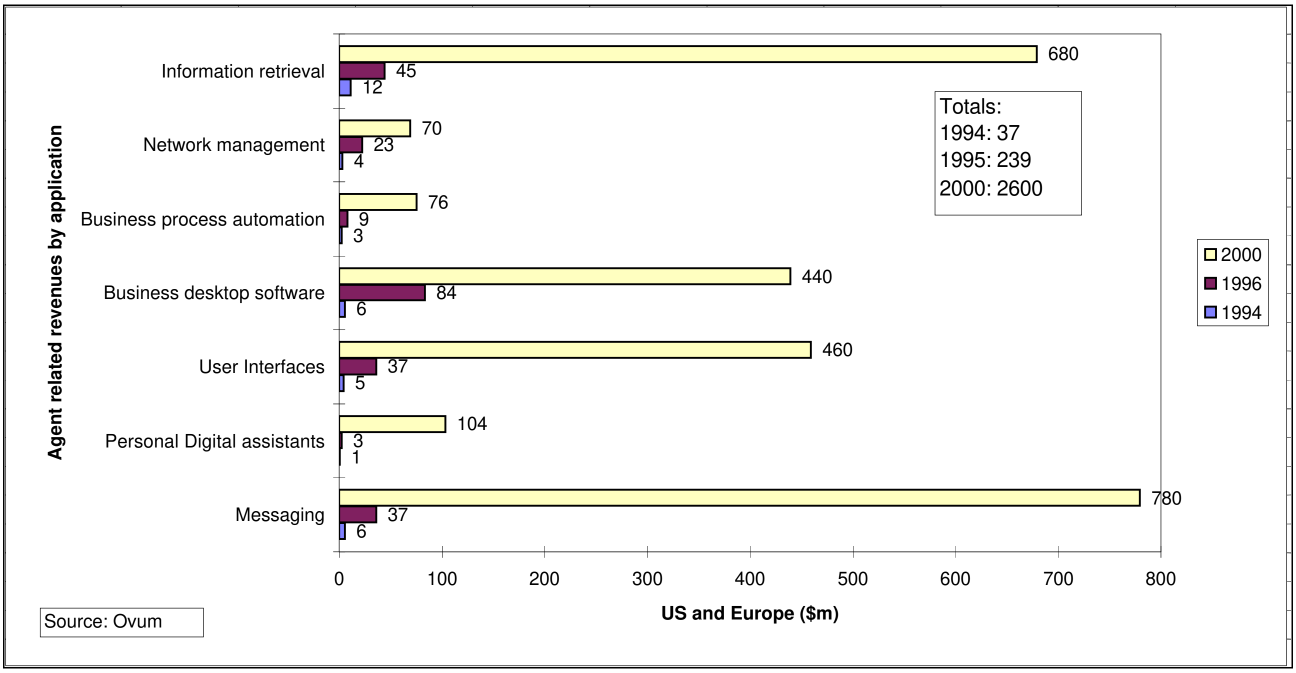
M. Wooldridge. 1995. This is MyWorld: The Logic of an Agent-Oriented DAI testbed. In Intelligent Agents - Theories, Architectures, and Languages (eds. M. Wooldridge and N. R. Jennings). Springer-Verlag Lecture Notes in Artificial Intelligence Volume 890. pp 263--274.

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

FIGURE CAPTIONS

Figure 1: Agent-related revenues by application

Sheet3



Information retrieval
Network management
Business process automation
Business desktop software
User Interfaces
Personal Digital assistants
Messaging
680
45
12
70
23
4
76
9
3
440
84
6
460
37
5
104
3
1
780
37
6
Totals:
1994: 37
1995: 239
2000: 2600
2000
1996
1994
Agent related revenues by application
0
100
200
300
400
500
600
700
800
US and Europe ($m)
Source: Ovum