Software Agents: An Overview

Hyacinth S. Nwana

Intelligent Systems Research
Advanced Applications & Technology Department
BT Laboratories, Martlesham Heath
Ipswich, Suffolk, IP5 7RE, U.K.
e-mail: hyacinth@info.bt.co.uk
Tel: (+44 1 473) 605457
fax: (+44 1 473) 642459

Knowledge Engineering Review, Vol. **11**, No 3, pp. 205-244, October/November 1996.

Abstract

Agent software is a rapidly developing area of research. However, the overuse of the word 'agent' has tended to mask the fact that, in reality, there is a truly heterogeneous body of research being carried out under this banner. This overview paper presents a typology of agents. Next, it places agents in context, defines them and then goes on, inter alia, to overview critically the rationales, hypotheses, goals, challenges and state-of-the-art demonstrators of the various agent types in our typology. Hence, it attempts to make explicit much of what is usually implicit in the agents literature. It also proceeds to overview some other general issues which pertain to all the types of agents in the typology. This paper largely reviews software agents, and it also contains some strong opinions that are not necessarily widely accepted by the agent community.

1 Introduction

The main goal of this paper is to overview the rapidly evolving area of software agents. The overuse of the word 'agent' has tended to mask the fact that, in reality, there is a truly heterogeneous body of research being carried out under this banner. This paper places agents in context, defines them and then goes on, *inter alia*, to overview critically the rationales, hypotheses, goals, challenges and state-of-the-art demonstrators/prototypes of the various agent types currently under investigation. It also proceeds to overview some other general issues which pertain to all the classes of agents identified. Finally, it speculates as to the future of the agents research in the short, medium and long terms. This paper largely reviews software agents. Since we are overviewing a broad range of agent types in this paper, we do not provide a definition of agenthood at this juncture. We defer such issues until Section 4 where we present our typology of agents.

The breakdown of the paper is as follows. Section 2 notes the situation of smart agents research in the broad field of Distributed Artificial Intelligence (DAI) and provides a brief history. Section 3 identifies the scope of applicability of agents research and notes that there is a diverse range of interested parties. Before the core critical overview of the agent typology of Section 5, Section 4 provides our view of what smart agents are; it also identifies the different types of agents which fall under the 'agents' banner and warns that *truly* smart or intelligent agents do not yet exist! They are still very much the aspiration of agent researchers. Section 6 overviews some more general issues on agents and and speculates briefly towards the future of agents in the short, medium and long terms. Section 7 concludes the paper.

2 Software Agents: History and the Context of this Paper

Software agents have evolved from multi-agent systems (MAS), which in turn form one of three broad areas which fall under DAI, the other two being Distributed Problem Solving (DPS) and Parallel AI (PAI). Hence, as with multi-agent systems, they inherit many of DAI's motivations, goals and potential benefits. For example, thanks to distributed computing, software agents inherit DAI's potential benefits including modularity, speed (due to parallelism) and reliability (due to redundancy). It also inherits those due to AI such as operation at the knowledge level, easier maintenance, reusability and platform independence (Huhns & Singh, 1994).

The concept of an agent, in the context of this paper, can be traced back to the early days of research into DAI in the 1970s - indeed, to Carl Hewitt's concurrent Actor model (Hewitt, 1977). In this model, Hewitt proposed the concept of a self-contained, interactive and concurrently-executing object which he termed 'actor'. This object had some encapsulated internal state and could respond to messages from other similar objects: an actor

> "is a computational agent which has a mail address and a behaviour. Actors communicate by message-passing and carry out their actions concurrently" (Hewitt, 1977, p. 131).

Broadly, for the purposes of this paper, we split the research on agents into two main strands: the first spanning the period 1977 to the current day, and the second from 1990 to the current day too. Strand 1 work on smart agents, which begun in the late 1970s and all through the 1980s to the current day, concentrated mainly on deliberative-type agents with symbolic internal models; later in this paper, we type these as collaborative agents. A deliberative agent is

> "one that possesses an explicitly represented, symbolic model of the world, and in which decisions (for example about what actions to perform) are made via symbolic reasoning" (Wooldridge, 1995, p. 42).

Initially, strand 1 work concentrated on *macro* issues such as the interaction and communication between agents, the decomposition and distribution of tasks, coordination and cooperation, conflict resolution via negotiation, etc. Their goal was to specify, analyse, design and integrate systems comprising of multiple collaborative agents. These resulted in classic systems and work such as the actor model (Hewitt, 1977), MACE (Gasser *et al.*, 1987), DVMT (Lesser & Corkill, 1981), MICE (Durfee & Montgomery, 1989), MCS (Doran *et al.*, 1990), the contract network coordination approach (Smith, 1980; Davis & Smith, 1983), MAS/DAI planning and game theories (Rosenschein, 1985; Zlotkin & Rosenschein, 1989; Rosenschein & Zlotkin, 1994). These 'macro' aspects of agents as Gasser (1991) terms them, emphasises the *society* of agents over *individual* agents, while *micro* issues relate specifically to the latter. In any case, such issues are well summarised in Chaib-draa *et al.* (1992), Bond & Gasser (1988) and Gasser & Huhns (1989). More recent work under this strand include TÆMS (Decker & Lesser, 1993; Decker, 1995) DRESUN (Carver *et al.*, 1991; Carver & Lesser, 1995), VDT (Levitt *et al.*, 1994), ARCHON (Wittig, 1992; Jennings *et al.*, 1995); Note that game theoretic work should arguably *not* be classed as a macro approach; it may, indeed, lie more towards the micro end of the spectrum.

In addition to the macro issues, strand 1 work has also been characterised by research and development into theoretical, architectural and language issues. In fact, such works evolve naturally, though not exclusively, from the investigation of the macro issues. This is well summarised in Wooldridge & Jennings (1995a), and in the edited collections of papers: Wooldridge & Jennings (1995b) and Wooldridge *et al.* (1996).

However, since 1990 or thereabouts, there has evidently been another distinct strand to the research and development work on software agents - *the range of agent types being investigated is now much broader*. Thus, this paper is complementary to Wooldridge & Jennings' (1995a) by placing emphasis on this strand although, naturally, there is some overlap, i.e. it overviews the broadening typology of agents being investigated by agent researchers. Some cynics may argue that this strand arises because everybody is now calling everything an agent, thereby resulting, inevitably, in such broadness. We sympathise with this viewpoint; indeed, it is a key motivation for this paper - to overview the extensive work that goes under the 'agent' banner. Essentially, our point is that in addition to investigating macro issues and others such as theories, architectures, languages, there has also been an unmistakable trend towards the investigation of a broader range of agent types or classes. The context of this paper is summarised in Table 1 below.

Strand 1	Macro issues	Bond & Gasser (1988) Gasser & Huhns (1989) Chaib-draa *et al.* (1992) Gasser *et al.* (1995)

	Theories, architectures & languages	Wooldridge & Jennings (1995a, 1995b) Wooldridge *et al.* (1996)
Strand 2	Diversification in the types of agents being investigated	*This paper covers this!*

Table 1 - Brief History of Software Agents and the Context of this Paper

3 Who are Investigating Software Agents for What and Why?

We eschew answering this question in a futuristic sense in favour of providing a flavour of the scope of the research and development underway in universities and industrial organisations. The range of firms and universities actively pursuing agent technology is quite broad and the list is ever growing. It includes small non-household names (e.g. Icon, Edify and Verity), medium-size organisations (e.g. Carnegie Mellon University (CMU), General Magic, Massachusetts Institute of Technology (MIT), the University of London) and the real big multinationals (e.g. Alcatel, Apple, AT&T, BT, Daimler-Benz, DEC, HP, IBM, Lotus, Microsoft, Oracle, Sharp). Clearly, these companies are by no means completely homogeneous, particularly if others such as Reuters and Dow Jones are appended to this list.

The scope of the applications being investigated and/or developed is arguably more impressive: it really does range from the mundane (strictly speaking, not agent applications) to the moderately 'smart'. Lotus, for example, will be providing a scripting language in their forthcoming version of Notes which would allow users to write their own individual scripts in order to manage their e-mails, calendars, and set up meetings, etc. This is based on the view that most people do not really need 'smart' agents. Towards the smart end of the spectrum are the likes of Sycara's (1995) *visitor hosting system* at CMU. In this system, "task-specific" and "information-specific" agents cooperate in order to create and manage a visitor's schedule to CMU. To achieve this, first, the agents access other on-line information resources in order to determine the visitor's areas of interest, name and organisation and resolve the inevitable inconsistencies and ambiguities. More information is later garnered including the visitor's status in her organisation and projects she is working on. Second, using the information gathered on the visitor, they retrieve information (e.g. rank, telephone number and e-mail address) from personnel databases in order to determine appropriate attendees (i.e. faculty). Third, the visitor hosting agent selects an initial list of faculty to be contacted, composes messages which it dispatches to the calendar agents of these faculties, asking whether they are willing to meet this visitor and at what time. If the faculty does not have a calendar agent, an e-mail is composed and despatched. Fourth, the responses are collated. Fifth, the visitor hosting agent creates the schedule for the visitor which involves booking rooms for the various appointments with faculty members. Naturally, the system interacts with the human organiser and seeks her confirmation, refutations, suggestions and advice.

Most would agree that this demonstrator is pretty smart, but its 'smartness' derives from the fact that the 'value' gained from individual stand-alone agents *coordinating* their actions by working in *cooperation*, is greater than that gained from any individual agent. This is *where* agents really come into their element.

More examples of applications are described later but application domains in which agent solutions are being applied to or investigated include workflow management, network management, air-traffic control, business process re-engineering, data mining, information

retrieval/management, electronic commerce, education, personal digital assistants (PDAs), e-mail, digital libraries, command and control, smart databases, scheduling/diary management, etc. Indeed, as Guilfoyle (1995) notes

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

The potential of agent technology has been much hailed, e.g. a 1994 report of Ovum's, a UK-based market research company, is titled "Intelligent agents: the new revolution in software" (Ovum, 1994). The same firm has apparently predicted that the market sector totals for agent software and products for USA and Europe will be worth at least $3.9 billion by the year 2000 in contrast to an estimated 1995 figure of $476 million (computed from figures quoted in Guilfoyle, 1995). Such predictions are perhaps overly optimistic.

Moreover, as King (1995) notes telecommunications companies like BT and AT&T are working towards incorporating smart agents into their vast networks; entertainment, e.g. television, and retail firms would like to exploit agents to capture our program viewing and buying patterns respectively; computer firms are building the software and hardware tools and interfaces which would harbour numerous agents; Reinhardt (1994) reports that IBM plans (or may have already done) to launch a system, the IBM Communications Systems (ICS), which would use agents to deliver messages to mobile users in the form they want it, be it fax, speech or text, depending on the equipment the user is carrying at the time, e.g. a PDA, a portable PC or a mobile phone. At BT Laboratories, we have also carried out some agent-related research on a similar idea where the message could be routed to the nearest local device, which may or may not belong to the intended recipient of the message. In this case, the recipient's agent negotiates with other agents for permission to use their facilities, and takes into consideration issues such as costs and bandwidth in such negotiations (Titmuss *et al*., 1996). At MIT, Pattie Maes' group is investigating agents that can match buyers to sellers or which can build coalitions of people with similar interests. They are also drawing from biological evolution theory to implement demonstrators in which some user only possesses the 'fittest' agents: agents would 'reproduce' and only the fittest of them will survive to serve their masters; the weaker ones would be purged.

It is important to note that most of these are still *demonstrators* only: converting them into real usable applications would provide even greater challenges, some of which have been anticipated but, currently, many are unforeseen. The essential message of this section is that agents are here to stay, not least because of their diversity, their wide range of applicability and the broad spectrum of companies investing in them. As we move further and further into the information age, any information-based organisation which does not invest in agent technology may be committing commercial hara-kiri.

4 What is an agent?

We have as much chance of agreeing on a consensus definition for the word 'agent' as AI researchers have of arriving at one for 'artificial intelligence' itself - nil! Recent postings to the software agents mailing list (agents@sunlabs.eng.Sun.COM) prove this. Indeed, in a couple of these postings, some propounded the introduction of a financial and/or legal aspect to the definition of agents, much to the derision of others. There are at least two reasons why it is so difficult to define precisely what agents are. Firstly, agent researchers do not 'own' this term in the same way as fuzzy logicians/AI researchers, for example, own the term 'fuzzy logic' - it is one that is used widely in everyday parlance as in travel agents, estate agents, etc.

Secondly, even within the software fraternity, the word 'agent' is really an umbrella term for a heterogeneous body of research and development. The response of some agent researchers to this lack of definition has been to invent yet some more synonyms, and it is arguable if these solve anything or just further add to the confusion. So we now have synonyms including knowbots (i.e. knowledge-based robots), softbots (software robot), taskbots (task-based robots), userbots, robots, personal agents, autonomous agents and personal assistants. To be fair, there are some good reasons for having such synonyms. Firstly, agents come in many physical guises: for example, those that inhabit the physical world, some factory say, are called robots; those that inhabit vast computer networks are sometimes referred to as softbots; those that perform specific tasks are sometimes called taskbots; and autonomous agents refer typically to mobile agents or robots which operate in dynamic and uncertain environments. Secondly, agents can play many roles, hence personal assistants or knowbots, which have expert knowledge in some specific domain. Furthermore, due to the multiplicity of roles that agents can play, there is now a plethora of adjectives which precede the word 'agent', as in the following drawn only from King's (1995) paper: search agents, report agents, presentation agents, navigation agents, role-playing agents, management agents, search and retrieval agents, domain-specific agents, development agents, analysis and design agents, testing agents, packaging agents and help agents. King's paper is futuristic and provides a *role-specific* classification of agents, and so such rampant metaphorical use of the word is fine. But there is also another view that it gives currency to others to refer to just about anything as an agent. For example, he considers "print monitors for open printing, fax redial, and others" (p. 18) as agents, albeit simple ones. As Wayner & Joch (1995) write, somewhat facetiously,

> "the metaphor has become so pervasive that we're waiting for some enterprising company to advertise its computer switches as *empowerment agents*" (p. 95).

We tend to use the word slightly more carefully and selectively as we explain later.

When we really have to, we define an agent as referring to a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user. Given a choice, we would rather say it is an umbrella term, meta-term or class, which covers a range of other more specific agent types, and then go on to list and define what these other agent types are. This way, we reduce the chances of getting into the usual prolonged philosophical and sterile arguments which usually proceed the former definition, when any old software is conceivably recastable as agent-based software.

4.1 A Typology of Agents

This section attempts to place *existing* agents into different agent classes, i.e. its goal is to investigate a typology of agents. A typology refers to the study of types of entities. There are several dimensions to classify existing software agents.

Firstly, agents may be classified by their mobility, i.e. by their ability to move around some network. This yields the classes of *static* or *mobile* agents.

Secondly, they may be classed as either *deliberative* or *reactive*. Deliberative agents derive from the deliberative thinking paradigm: the agents possess an internal symbolic, reasoning model and they engage in planning and negotiation in order to achieve coordination with other agents. Work on reactive agents originate from research carried out by Brooks (1986) and Agre & Chapman (1987). These agents on the contrary do not have any internal, symbolic models of their environment, and they act using a stimulus/response type of

behaviour by responding to the present state of the environment in which they are embedded (Ferber, 1994). Indeed, Brooks has argued that intelligent behaviour can be realised without the sort of explicit, symbolic representations of traditional AI (Brooks, 1991b).

Thirdly, agents may be classified along several ideal and primary attributes which agents *should* exhibit. At BT Labs, we have identified a minimal list of three: autonomy, learning and cooperation. We appreciate that any such list is contentious, but it is no more or no less so than any other proposal. Hence, we are not claiming that this is a necessary or sufficient set. *Autonomy* refers to the principle that agents can operate on their own without the need for human guidance, even though this would sometimes be invaluable. Hence agents have individual internal states and goals, and they act in such a manner as to meet its goals on behalf of its user. A key element of their autonomy is their proactiveness, i.e. their ability to 'take the initiative' rather than acting simply in response to their environment (Wooldridge & Jennings, 1995a). *Cooperation* with other agents is paramount: it is the *raison d'être* for having multiple agents in the first place in contrast to having just one. In order to cooperate, agents need to possess a social ability, i.e. the ability to interact with other agents and possibly humans via some communication language (Wooldridge & Jennings, 1995a). Having said this, it is possible for agents to coordinate their actions without cooperation (Nwana *et al.*, 1996). Lastly, for agent systems to be truly 'smart', they would have to *learn* as they react and/or interact with their external environment. In our view, agents are (or should be) disembodied bits of 'intelligence'. Though, we will not attempt to define what intelligence is, we maintain that a key attribute of any intelligent being is its ability to learn. The learning may also take the form of increased performance over time. We use these three minimal characteristics in Figure 1 to derive four types of agents to include in our typology: *collaborative agents, collaborative learning agents, interface agents* and truly *smart agents*.

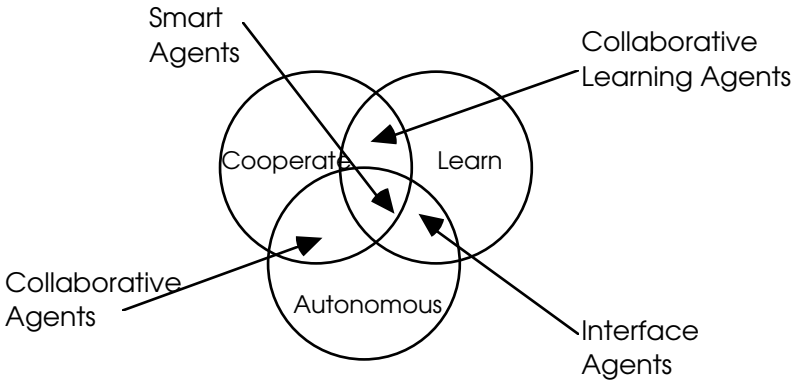

Figure 1 - A Part View of an Agent Typology

We emphasise that these distinctions are *not* definitive. For example, with collaborative agents, there is more emphasis on cooperation and autonomy than on learning; hence, we do not imply that collaborative agents never learn. Likewise, for interface agents, there is more emphasis on autonomy and learning than on cooperation. We do *not* consider anything else which lie outside the 'intersecting areas' to be agents. For example, most expert systems are largely 'autonomous' but, typically, they do not cooperate or learn. Ideally, in our view, agents should do all three equally well, but this is the *aspiration* rather than the reality. Truly smart agents do not yet exist: indeed, as Maes (1995a) notes "current commercially available agents barely justify the name", yet alone the adjective 'intelligent'. Foner (1993) is even more incandescent; though he wrote this in 1993, it still applies today:

> "... I find little justification for most of the commercial offerings that call themselves agents. Most of them tend to excessively anthromomorphize the software, and then conclude that it must be an agent because of that very anthropomorphization, while simultaneously failing to provide any sort of discourse or "social contract" between the user and the agent. Most are barely autonomous, unless a regularly-scheduled batch job counts. Many do not degrade gracefully, and therefore do not inspire enough trust to justify more than trivial delegation and its concomitant risks" (Foner, 1993, 39/40).

In effect, like Foner, we assert that the arguments for most commercial offerings being agents suffer from the logical fallacy of *petitio principii* - they assume what they are trying to prove - or they are circular arguments. Indeed, this applies to other 'agents' in the literature.

In principle, by combining the two constructs so far (i.e. static/mobile and reactive/deliberative) in conjunction with the agent types identified (i.e. collaborative agents, interface agents, etc.), we could have *static deliberative collaborative agents, mobile reactive collaborative agents, static deliberative interface agents, mobile reactive interface agents*, etc. But these categories, though quite a mouthful, may also be necessary to further classify existing agents. For example, Lashkari *et al.* (1994) presented a paper at AAAI on 'Collaborative interface agents' which, in our classification, translates to *static collaborative interface agents*.

Fourthly, agents may sometimes be classified by their roles (preferably, if the roles are major ones), e.g. world wide web (WWW) information agents. This category of agents usually exploits internet search engines such as WebCrawlers, Lycos and Spiders. Essentially, they help manage the vast amount of information in wide area networks like the internet. We refer to these class of agents in this paper as *information* or *internet agents*. Again, information agents may be static, mobile or deliberative. Clearly, it is also pointless making classes of other minor roles as in report agents, presentation agents, analysis and design agents, testing agents, packaging agents and help agents - or else, the list of classes will be large.

Fifthly, we have also included the category of *hybrid* agents which combine of two or more agent philosophies in a single agent.

There are other attributes of agents which we consider *secondary* to those already mentioned. For example, is an agent versatile (i.e. does it have many goals or does it engage in a variety of tasks)? Is an agent benevolent or non-helpful, antagonistic or altruistic? Does an agent lie knowingly or is it always truthful (this attribute is termed veracity)? Can you trust the agent enough to (risk) delegate tasks to it? Is it temporally continuous? Does it degrade gracefully in contrast to failing drastically at the boundaries? Perhaps unbelievably, some researchers are also attributing emotional attitudes to agents - do they get 'fed up' being asked to do the same thing time and time again? What role does emotion have in constructing believable agents (Bates, 1994)? Some agents are also imbued with *mentalistic* attitudes or notions such as beliefs, desires and intentions - referred to typically as BDI agents (Rao & Georgeff, 1995). Such attributes as these provide for a stronger definition of agenthood.

In essence, *agents exist in a truly multi-dimensional space*, which is why we have not used a two or three-dimensional matrix to classify them - this would be incomplete and inaccurate. However, for the sake of clarity of understanding, we have 'collapsed' this multi-dimensional space into a single list. In order to carry out such an audacious move, we have made use of our knowledge of the agents we know are currently 'out there' and what we wish to aspire to. Therefore, the ensuing list is to some degree arbitrary, but we believe these types cover most of the agent types being investigated currently. We have left out collaborative learning agents, see Figure 1, on the grounds that we do not know of the existence 'out there' of any

such agents which collaborate and learn, but are not autonomous. Hence, we identify seven types of agents:

- Collaborative agents
- Interface agents
- Mobile agents
- Information/Internet agents
- Reactive agents
- Hybrid agents
- Smart Agents

There are some applications which combine agents from two or more of these categories, and we refer to these as *heterogeneous agent systems*. Such applications already exist even though they are relatively few. However, we also overview briefly such systems in the next section.

Another issue of note (for completeness sake) is that agents need not be benevolent to one another. It is quite possible that agents may be in competition with one another, or perhaps quite antagonistic towards each other. However, we view *competitive* agents as *potential* subclasses of all these types. That is, it is possible to have competitive collaborative-type agents, competitive interface agents, competitive information agents, etc.

4.2 A Critique of Our Typology

As with our definition of agenthood, our typology of agents is bound to be contentious. Two official reviewers of this paper all took issue with it, but their suggestions are, in our opinion, either more debatable or unclear. One reviewer, reviewer 1, claimed that we have confused agents that are defined by what they *do* (information agents, interface agents and collaborative agents), and other types for the sort of *technology* that underpins these agents (mobile agents, reactive agents, hybrid agents). Thus, he/she would have preferred a 2-dimensional classification. The second reviewer mentioned a similar point but alluded to a different classification. To a large degree, we disagree with this criticism, though not fully. We believe we had already attempted, perhaps unsuccessfully, to pre-empt this criticism. Firstly, we would not group information agents, interface agents and collaborative agents in one large group: in our view, as we explained earlier, collaborative agents and interface agents are defined by what they *are*, while information agents are defined by what they *do*. Secondly, we do not agree fully with the assertion that mobile agents, reactive agents and hybrid agents are all underlying technologies for implementing the former classes. To this reviewer, interface agents are collaborative agents implemented using reactive technology! We simply disagree with this viewpoint. As we explain later in the paper, reactive agents for example, have a distinct philosophy, hypothesis, etc. which make it stand out from the rest. We have surveyed the area of technologies for building software agent systems in another paper, Nwana & Wooldridge (1996). However, we agree with the general thrust of the argument to some degree; for example, we fully accept the reviewers' viewpoint that mobility is not a necessary condition for agenthood - a point which is implicit in Section 4.1, and which we explain later. Thirdly, we address such issues when we discuss the individual types more fully in the rest of the paper. Fourthly, we point out, explicitly, in Section 4.1 that agents exist in a truly multi-dimensional space, and that for the sake of *clarity of*

understanding, we have collapsed this multi-dimensional space into a single list. To produce this list, we used a set of criteria which included inate properties of agents which we would prefer to see (autonomy, cooperation, learning), other constructs (static/mobile, deliberative/reactive), major roles (as in information agents) and whether they are hybrid or heterogeneous. In a previous version of this paper where we had a more hierarchical breakdown, it turned out to be less clear. Fifthly, other typologies in the literature are equally as contentious. For example, Wooldridge & Jennings (1995a) broadly classify agents into the following: gopher agents, service performing agents and proactive agents. We believe this is too general and simplistic a classification. It is for these reasons that we opted for such a 'flat' breakdown. To be fair, apart from the typology, these two reviewers were *very* complementary about the paper.

In conclusion, our typology is not without its critics (but so are all others), but as reviewer 1 pointed out "while I agree that most agents in the literature can be categorised into these types, I think the types are themselves faulty". In this paper, we have deliberately traded in accuracy for clarity. Our typology highlights the key *contexts* in which the word 'agent' is used in the software literature.

4.3 What Agents are Not

In general, we have already noted that a software component which does not fall in one of the intersecting areas of Figure 1 does not count as an agent. In any case, before the word 'agent' came into vogue within the computing/AI fraternity, Minsky, in his *Society of Mind* (1985), had already used it to formulate his theory of human intelligence. However, Minsky used it to refer to much more *basic* entities:

> ". .to explain the mind, we have to show how minds are built from mindless stuff, from parts that are much smaller and simpler than anything we'd consider smart... But what could those simpler particles be - the "agents" that compose our minds? This is the subject of our book..." (Minsky, 1985, 18).

Clearly, Minsky's use of the word 'agent' is quite distinct from its use in this paper.

Furthermore, as noted earlier, expert systems do not meet the preconditions of agenthood, as do most knowledge-based system applications. Modules in distributed computing applications do not constitute agents either as Huhns & Singh (1994) explain. First, such modules are rarely 'smart', and hence much less robust than agents are (or should be); they also do not degrade gracefully. Second, in agent-based systems generally, the communication involves involves high-level messages in contrast to the low-level messaging in distributed computing. The use of high-level messaging leads to lower communication costs, easy re-implementability and concurrency. Lastly, and perhaps most importantly, agent-based applications operate typically at the *knowledge level* (Newell, 1982), not at the *symbol level* as is the case in distributed computing applications. In any case, modules in distributed computing applications are not autonomous in the same sense as described earlier for agent applications. The majority of software applications may be ruled out from the set of agent-based applications on the same grounds that expert systems or distributed computing applications are.

5 A Panoramic Overview of the Different Agent Types

In this section we, in turn, overview all the types of agents identified in our typology of the previous section bar smart agents. Figure 2 summarises these types and lists the order in

which they are surveyed. In particular, we would overview them in terms of *some* or *all* of following: their essential metaphors, hypotheses/goals, motivations, roles, prototypical examples, potential benefits, key challenges, and some other general issues about the particular agent type. We do not overview the shaded type, smart agents, on the grounds that this is the *aspiration* of agent researchers rather than the *reality*.

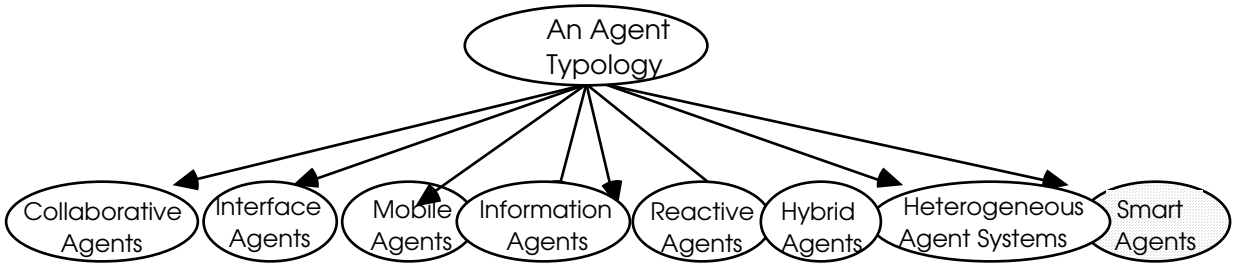

Figure 2 - A Classification of Software Agents

5.1 Collaborative Agents: An Overview

As shown in Figure 1, collaborative agents emphasise autonomy and cooperation (with other agents) in order to perform tasks for their owners. They may learn, but this aspect is not typically a major emphasis of their operation. In order to have a coordinated set up of collaborative agents, they may have to *negotiate* in order to reach mutually acceptable agreements on some matters. Most of the work classified in this paper as strand 1 (see Section 2) investigated this class of agents. As noted earlier, some AI researchers are providing stronger definitions to such agents, e.g. some attribute mentalistic notions such as beliefs, desires and intentions - yielding BDI-type collaborative agents. Hence, the class of collaborative agents may itself be perceived as a broad church.

In brief the key general characteristics of these agents include autonomy, social ability, responsiveness and proactiveness. Hence, they are (or should/would be) able to act rationally and autonomously in open and time-constrained multi-agent environments. They tend to be static, large coarse-grained agents. They may be benevolent, rational, truthful, some combination of these or neither. Typically, most currently implemented collaborative agents do not perform any complex learning, though they may or may not perform limited parametric or rote learning.

5.1.1 Hypothesis/Goal

The *hypothesis*, rationale or goal for having collaborative agent systems is a specification of the goal of DAI as noted in Huhns & Singh (1994). Paraphrasing these authors, it may be stated as ‘creating a system that interconnects separately developed collaborative agents, thus enabling the ensemble to function beyond the capabilities of any of its members’. Formally,

$$V(\_agent_i) > \max(V(agent_i))$$

where V represents ‘value addedness’. This could have an arbitrary definition involving attributes such as speed, worst-case performance, reliability, adaptability, accuracy or some combination of these.

5.1.2 Motivation

The *motivation* for having collaborative agent systems may include one or several of the following (they are a specialisation of the motivations for DAI):

- To solve problems that are too large for a centralised single agent to do due to resource limitations or the sheer risk of having one centralised system;
- To allow for the interconnecting and interoperation of multiple existing legacy systems, e.g. expert systems, decision support systems, etc.;
- To provide solutions to inherently distributed problems, e.g. distributed sensor networks (cf. DVMT, Durfee *et al.,* 1987) or air-traffic control;
- To provide solutions which draw from distributed information sources, e.g. for distributed on-line information sources, it is natural to adopt a distributed and collaborative agent approach;
- To provide solutions where the expertise is distributed, e.g. in health care provisioning;
- To enhance modularity (which reduces complexity), speed (due to parallelism), reliability (due to redundancy), flexibility (i.e. new tasks are composed more easily from the more modular organisation) and reusability at the knowledge level (hence shareability of resources);
- To research into other issues, e.g. understanding interactions among human societies.

5.1.3 A Prototypical Example: CMU's Pleiades System

The Pleiades project at CMU directed by Tom Mitchell and Katia Sycara has, as one of its objectives, to investigate methods for automated negotiation among *collaborative* [1] agents, in order to improve their robustness, effectiveness, scalability and maintainability (see http: URL1). The project applies collaborative agents in the domain of Organisational Decision Making over the "InfoSphere" (which refers essentially to a collection of internet-based heterogeneous resources). This infosphere is ripe for the application of these class of agents not least because it is inherently a distributed set of on-line information sources.

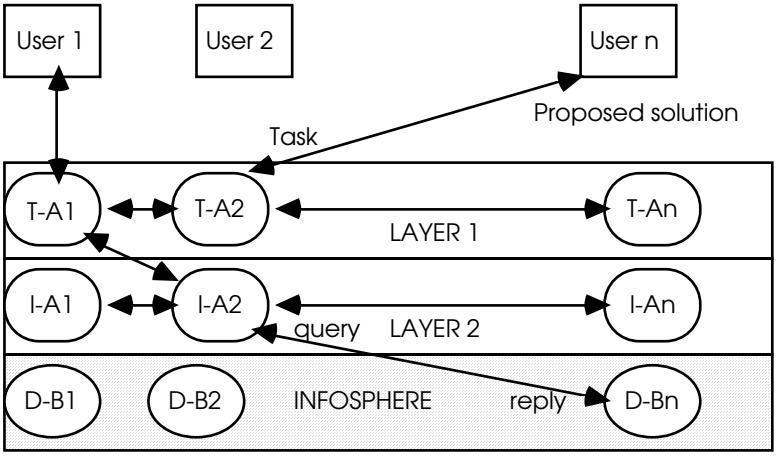

Figure 3 - The Pleiades Distributed System Architecture (Adapted from Sycara, 1995)

[1] My interpretation and emphasis.

Pleiades is a distributed collaborative agent-based architecture which has two layers of abstraction: the first layer which contains *task-specific* collaborative agents and second layer which contains *information-specific* collaborative agents (see Figure 3). This architecture was used to develop the visitor hosting system which was described briefly earlier. Task-specific agents, depicted as task-assistants (T-A) in the figure, perform a particular task for its user, e.g. arranging appointments and meetings with other task-specific agents. These agents coordinate and schedule plans based on the context. They collaborate with one another (at level 1) in order to resolve conflicts or integrate information. In order to garner the information required at this level, they request information from information-specific agents, depicted as information assistants (I-A) in Figure 3. Information-specific agents, in turn, may collaborate with one another (i.e. within layer 2) in order to provide the information requested back to the layer 1 requesting agent. The source of the information are the many databases (D-B) in the infosphere. Ultimately, the task agent proposes a solution (sometimes an intermediate one) to its user.

Task-specific agents have the following knowledge (Sycara, 1995): a model of the task domain, knowledge of how to perform the task, knowledge of how to gather the information for the task, knowledge of other task-specific or information-specific agents it must coordinate with in order to meet the task, protocols that enable coordination with other agents and, lastly, strategies for conflict resolution and information fusion. They also possess some learning mechanisms, e.g. when an agent needs to learn the preferences of its user. On the contrary, information-specific agents know of the following: knowledge of the databases that it is associated with (in addition to details such as their size, average time it takes to answer a query and monetary costs for query processing), knowledge of how to access the databases, knowledge of how to resolve conflicts and information fusion strategies, and protocols for coordination with other relevant software agents. These agents are also ‘smart’ enough to cache answers to frequently asked queries, and can also induce database regularities which they use during inter-agent interactions.

The main rationale of the architecture is to provide software agents for retrieving, filtering and fusing information from distributed, multi-modal sources and that the agents should assist in decision making. Sycara and her colleagues hypothesised that in order to meet their goals, they would need a distributed collection of collaborative agents which can gather, filter and fuse information in addition to being able to learn from their interactions. Agents communicate using KQML (Finin & Wiederhold, 1991) and e-mail, and they negotiate in order to reach agreements in cases of conflicts. The layered architecture is clearly very modular, indeed modular enough for Sycara and her colleagues to have introduced connectionist modules in the design of other systems.

Clearly, there is much sophistication to this architecture even though we have left out much more interesting details (e.g. *how* and *what* the agents learn). Individually, an agent consists of a planning module linked to its local beliefs and facts database. It also has a local scheduler, a coordination module and an execution monitor. Thus, agents can instanstiate task plans, coordinate these plans with other agents and schedule/monitor the execution of its local actions. Interestingly, the architecture has no central planner and hence agents must all engage in coordination by communicating to others their constraints, expectations and other relevant information.

The Pleiades architecture shows clearly how collaborative agents can operate in concert such that their ensemble functions beyond the capabilities of any individual agent in the set-up.

Apart from the visitor hosting system, other systems have also been developed using this architecture/methodology in the domains of financial portfolio management, emergency medical care and electronic commerce (Sycara, 1995). In addition to these domains, there are others ripe for exploiting collaborative agents including workflow management, network management and control, telecommunication networks and business process engineering. In all these domains, collaborative agents may provide much 'added value' to current single agent-based applications.

5.1.4 A Brief Critical Review of Collaborative Agent Systems Work

There are many other useful pieces of work on collaborative agents. The first important point to re-emphasise is the fact that much work classified in this paper as strand 1 work (see Section 2) exploited collaborative, deliberative agents; they may not have been fully collaborative as defined in this paper, but they were in spirit. For example, each agent in Durfee *et al.*'s (1987) distributed vehicle monitoring system (DVMT) is a blackboard knowledge source whose task is to identify the vehicle's track from acoustic data. Each of these agents shared a global knowledge of the problem solving; hence, they are, strictly speaking, not that autonomous and cooperation is quite basic as it all proceeds via the common blackboard. Certainly, other strand 1 testbeds such as MACE (Gasser *et al*., 1987), MCS (Doran *et al*., 1990) and IPEM (Ambros-Ingerson & Steel, 1988) have deliberative agents with planning modules that underpin the coordination and cooperation in the set-ups they are operating within. In the case of IPEM and MCS, non-linear planners are used. Other planning-based prototypes include Hayes-Roth's (1991) GUARDIAN architecture and Cohen's *et al.*'s PHOENIX system. At BT Labs, two prototype collaborative agent-based systems have been developed recently: the ADEPT and MII prototypes. ADEPT (O'Brien & Wiegand, 1996) employs collaborative agents in the application area of business process re-engineering while MII (Titmuss *et al*., 1996) demonstrates that collaborative agents can be used to perform decentralised management and control of consumer electronics, typically PDAs or PCs integrated with services provided by the network operator.

As regards collaborative agents with much stronger definitions, mention has earlier been made of Rao & Georgeff's characterisation of rational agents in terms of the mental attitudes of beliefs, desires and intentions (Rao & Georgeff, 1992). These are the attitudes typical of *epistemic* logics. Such work on stronger definitions of collaborative agents is much in progress. Another useful piece of research which attributes mentalistic notions to collaborative agent-based system design is Shoham's (1993) work on agent-oriented programming. In this work, an agent's mental state is described by its beliefs, decisions, capabilities and obligations, and Shoham's language introduces epistemic and *deontic* modal operators for such notions. This is because in order for agents to reason about these mentalistic attitudes, logics and operators for describing them must be developed. Other agent frameworks based on such mentalistic attitudes include Bratman *et al.*'s (1988) IRMA and Jennings' (1993) GRATE/GRATE* environments. Much of the latter work exploits Cohen & Levesque's (1990) classic work.

The key criticism of collaborative agents levelled by some researchers stems from their grounding in the deliberative thinking paradigm which has dominated AI research over the last thirty years. Some researchers, particularly those in the reactive agents camp, believe that intelligent behaviour can be generated without the sort of explicit symbolic-level

representations (and hence, reasoning) prevalent in AI (e.g. Brooks, 1991b). That is, they object to agents having an internal representation of actions, goals and events required by the planning module to determine the sequence of actions that will achieve the goals. Researchers like Agre & Chapman (1987) have challenged the usefulness of having elaborate plans; they argue that a rational, goal-directed activity need not be organised as a plan. They concede that people use plans, but they argue that in real life there is much moment-to-moment *improvisation* with any plan, which is dependent on the 'situation' of the relevant agent in its physical and social world. Clearly, though this criticism is targeted at the entire deliberative school of AI, it also impacts on deliberative, collaborative agents of whatever complexion. Hence, they contend that such deliberative agents would result in brittle and inflexible demonstrators with slow response times. This viewpoint led to birth of the reactive agents paradigm based on situated-action theory discussed later.

As regards stronger collaborative agent definitions (e.g. BDI agents), Rao & Georgeff (1995) acknowledge the two main criticisms levelled at such work as theirs and Bratman *et al.*'s (1988). First, while traditional planning researchers and classical decision theorists question the necessity for having all of these epistemic attitudes (i.e. beliefs, desires, intentions), DAI researchers with a sociological bias question why they only have three! Secondly, the logics underpinning these agents, mostly second-order modal logics, have not been investigated fully and their relevance in practice is questionable. Rao & Georgeff (1995) tackle both these issues in their paper. Indeed, they proceed to describe how BDI agents, with some simplifying assumptions to their theoretical framework, are being applied to large-scale applications - in this case, OASIS, an air-traffic management application prototype which has been successfully tested at Sydney airport in 1995. This prototype has been tested with 100+ aircraft agents and 10+ global agents which handle other issues including windfields, trajectories and coordination (Georgeff, 1996). Full implementation is already in progress. However, it must be emphasised that research into such stronger definition of agents, relatively, is still very much in its infancy.

Some more 'criticisms' of collaborative agents are presented next as challenges still to be addressed by collaborative agent researchers.

5.1.5 Collaborative Agents: Some Key Challenges

Despite successful demonstrators like the Pleiades system and MII (Titmuss *et al.*, 1996), these agents have been deployed in none but a few real industrial settings though this situation is changing, e.g. those built under the auspices of the ARCHON project (Wittig, 1992; Jennings *et. al*., 1993) or a couple of others built with the involvement of Mike Georgeff, e.g. the Space Shuttle Malfunction Handling system and the agent-based Royal Australian Airforce Simulator (Georgeff, 1996). There are still many teething problems; we mention several here. Note that these are not necessarily specific to collaborative agents only:

- Engineering the construction of collaborative agent systems: the Pleiades system is a good step in this direction but there is much more research to do. To paraphrase BT's Prof. Robin Smith, we must move away from point solutions to point problems, and design methodologies/meta-tools which allow for quicker implementation of collaborative agent-based systems (Smith, 1996b);

- Inter-agent coordination: this is a major issue in the design of these systems. Coordination is essential to enabling groups of agents to solve problems effectively. Without a clear theory of coordination, anarchy or deadlock can set in easily in collaborative agent systems? Furthermore, should agents be totally truthful when

negotiating with others or should they be allowed to 'lie' when it suits them? Coordination is also required due to the constraints of resource boundedness and time. Much experimental and/or formal work is still required to address these issues of coordination and negotiation.

- Stability, Scalability and Performance Issues: these issues have yet to be acknowledged, yet alone tackled in collaborative agent systems research. Empirical investigations need to be carried out to establish suitable minimum levels of performance and, clearly, these systems have to be stable. Alternatively, their stabilities would need to be proven formally. Though, these issues are *non-functional*, they are crucial nonetheless;
- Legacy systems: the thorny issue of what to do with legacy systems is still with us and will always be a problem. Established techniques and methodologies for integrating agents and legacy systems are still required;
- How do these systems learn? Would learning not lead to instability? What are the appropriate architectures for different types of problems? How do you ensure an agent does not spend much of its time learning, instead of participating in its set-up?
- Evaluation of collaborative agent systems: this problem is still outstanding. How are they verified and validated to ensure they meet their functional specifications? Are unanticipated events handled properly? How else would you trust such systems to run power stations, nuclear installations and chemical plants?

In conclusion, despite the criticisms of collaborative agents by those within and without other agent camps, there are many industrial applications which would benefit significantly from them, in just the same way as there are applications which would benefit from reactive agents. For example, at BT, we see a potential major role for them in managing telecommunications networks and in business process management (Nwana, 1996).

5.2 Interface Agents: An Overview

Interface agents (c.f. Figure 1) emphasise autonomy and learning in order to perform tasks for their owners. Pattie Maes, a key proponent of this class of agents, points out that the key metaphor underlying interface agents is that of a *personal assistant* who is *collaborating with the user* in the same work environment. Note the subtle emphasis and distinction between collaborating with *the user* and collaborating with *other agents* as is the case with collaborative agents. Collaborating with a user may not require an explicit agent communication language as one required when collaborating with other agents.

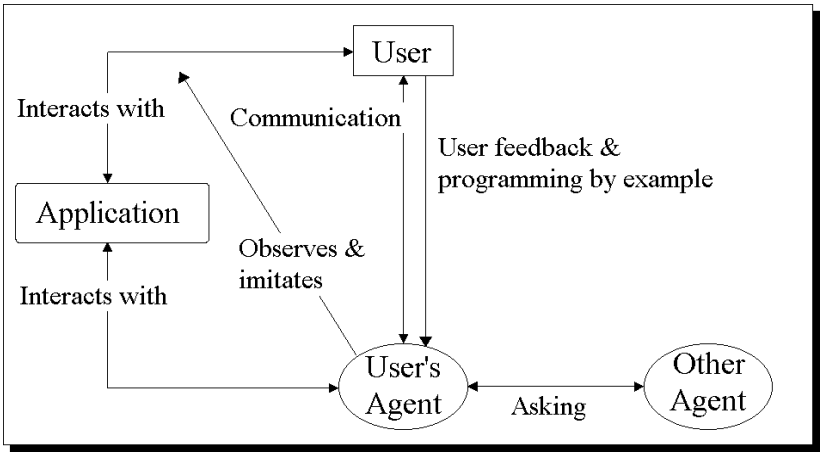

Figure 4 - How Interface Agents Work (adapted from Maes, 1994)

Figure 4 depicts the functioning of interface agents. Essentially, interface agents support and provide assistance, typically to a user learning to use a particular application such as a spreadsheet or an operating system. The user's agent observes and monitors the actions taken by the user in the interface (i.e. 'watches over the shoulder of its user'), learns new 'short-cuts', and suggests better ways of doing the task. Thus, the user's agent acts as an autonomous personal assistant which cooperates with the user in accomplishing some task in the application. As for learning, interface agents learn typically to better assist its user in four ways (Maes, 1994) all shown in Figure 4:

- By observing and imitating the user (i.e. learning from the user);
- Through receiving positive and negative feedback from the user (learning from the user);
- By receiving explicit instructions from the user (learning from the user);
- By asking other agents for advice (i.e. learning from peers).

Their cooperation with other agents, if any, is limited typically to asking for advice, and not in getting into protracted negotiation deals with them as is the case with collaborative agents. The learning modes are typically by rote (memory-based learning) or parametric, though other techniques such as evolutionary learning are also being introduced. In summary a learning interface agent,

> "as opposed to any kind of agent, is one that uses machine-learning techniques to present a pseudo "intelligent" user interface for its actions" (Foner, 1993, p. 1).

5.2.1 Hypothesis/Goal

The objective of interface agents research (as Maes sees it) is to work towards Alan Kay's dream of having indirectly managed human-computer interfaces (Kay, 1990). The argument goes as follows. Current computer user interfaces only respond to direct manipulation, i.e. the computer is passive and always waits to execute highly specified instructions from the user. It provides little or no *proactive* help for complex tasks or for carrying out actions such as searches for information that may take an indefinite time (Maes, 1995). In the future, there will be millions of untrained users attempting to make use of computers and networks of tomorrow. Therefore, instead of a user issuing direct commands to some interface, he could be engaged in cooperative process in which human and software agents can both initiate communication, perform tasks and monitor events. This cooperation between human and agent would benefit the human in using this application.

Hence, the *goal* is to migrate from the direct manipulation metaphor to one that delegates some of the tasks to (proactive and helpful) software interface agents in order to accommodate novice users. The *hypothesis* is that these agents can be trusted to perform competently some tasks delegated to them by their users. More specifically, that

> "under certain conditions, an interface agent can "program itself" (i.e., it can acquire the knowledge it needs to assist its user). The agent is given a minimum of background knowledge, and it learns appropriate "behavior" from the user and from other agents" (Maes, 1994, p. 89).

She goes on to explain that two preconditions need to be fulfilled by suitable application domains: firstly, that there is substantial repetitive behaviour in using the application

(otherwise, the learning agent will not be able to learn anything) and, secondly, that this repetitive behaviour is potentially different for different users (otherwise, use a knowledge-based approach).

5.2.2 Motivation

To recap, an interface agent is a quasi-smart piece of software which assists a user when interacting with one or more computer applications. Therefore, the *motivating*, underlying principle of interface agents seems to be that there is no inherent merit in drudgery. Where boring and laborious tasks (particularly, but not exclusively at the user interface) could be delegated to interface agents, they should be - in order to eliminate the tedium of humans performing several manual sub-operations say. A motivating reason for the choice of domains that Maes' group has tackled has been their dissatisfaction with the ways that tasks in these domains are handled currently. For example, she explains that valuable hours are wasted managing junk mail, scheduling and rescheduling meetings, searching for information among heaps of it, etc. - indeed, the title of her 1994 paper captures succinctly her motivation: 'Agents that Reduce Work and Information Overload' (Maes, 1994).

5.2.3 Benefits/Roles

The general benefits of interface agents are threefold. First, they make less work for the end user and application developer. Secondly, the agent can adapt, over time, to its user's preferences and habits. Finally, know-how among the different users in the community may be shared (e.g. when agents learn from their peers). Perhaps these will be understood better by discussing some of the roles for which Maes and her team at MIT are building interface agents. Thus far, her team has constructed demonstrator agents for the following roles:

- Eager assistants (e.g. Kozierok & Maes, 1993);
- Guides (e.g. Liebermann, 1995);
- Memory aids (e.g. Rhodes & Starner, 1996);
- Filter/critics (e.g. Sheth & Maes, 1993);
- Matchmaking/referrals (e.g. Foner, 1996);
- Buying/selling on your behalf (e.g. Chavez & Maes, 1996);
- Entertainment (e.g. Maes, 1995b).

We overview them briefly.

Kozierok & Maes (1993) describe an interface agent, Calendar Agent, for scheduling meetings which is attachable to any application provided it is scriptable and recordable, e.g. scheduling software package. Calendar Agent assists (i.e. its role is in *assisting*) its user in scheduling meetings which involves accepting, rejecting, scheduling, negotiating and rescheduling meeting times. It really comes into its element because it can learn, over time, the preferences and commitments of its user, e.g. she does not like to attend meetings on Friday afternoons, he prefers meetings in the morning, etc. The learning techniques employed are memory-based learning and reinforcement learning.

Liebermann (1995) describes an agent called Letizia (a keyword and heuristic-based search agent) which assists in web browsing. Letizia's role is that of a *guide*. When users operate their favourite browser, e.g. Netscape, they must state their interests explicitly when using

traditional search engines such as Webcrawler or Lycos. The user remains idle while the search is in progress, and likewise, the search engine is idle whilst the user is browsing the interface. Essentially, Letizia provides a cooperative search between itself and the user. Since most browsers encourage depth-first browsing, Letizia conducts a breadth-first search concurrently for other useful locations that the user may be interested in. It does this by 'guessing' the user's intention and proceeding to search using the search engine. It guesses the user's intentions via inferring from his/her browsing behaviour, e.g. she keeps returning to some particular page, you enter a page into your hotlist or you download some article. The user's actions immediately refocus the search. By doing this, it is able to recommend some other useful serendipitous locations.

The Remembrance Agent (Rhodes & Starner, 1996) is attached currently to an Emacs editor. As the user composes some e-mail message, say, the agent is able to carry out a keyword search and retrieve the five most relevant e-mails in her directory relating to this e-mail being composed. It is really successful when it recommends continuously and unobstrusively invaluable documents, e-mails or files which you would otherwise have forgotten when, for example, you are composing some new document. It can also be used, conceivably when browsing the web or writing a paper; in the latter case, the remembrance agent may recommend other researchers' papers which should be consulted. Hence, its role is clearly that of a *memory* aid.

Sheth & Maes (1993) and Maes (1994) describe a news filtering agent, called NewT, whose role is that of helping the user *filter* and select articles from a continuous stream of Usenet Netnews. The idea is to have the user create one or many "news agents" (e.g. one agent for sports news, one for financial news, etc.) and train them by example (i.e. by presenting to them positive and negative examples of what should/or should not be retrieved). It is message-content, keyword-based but it also exploits other information such as the author and source. NewT is even more complicated because a user's population of information filtering agents evolve with time using genetic computing techniques. Indeed, some similar new work at MIT is investigating agents that 'breed' in their environment, i.e. information agents, given feedback on the information returned, breeds progressively more of those which return 'good quality' information, and purges the rest that do not (see Moukas, 1996).

Foner (1996) reports on his Yenta/Yenta-lite *matchmaking* agent prototype whose goals scenarios include being able to match buyers and sellers of some item and introducing them to one another, finding and grouping people with compatible professional or personal interests, or building coalitions of people interested in the same topics. Each user in the community has a Yenta agent. Yenta agents are able to carry out *referrals* which work in the same fashion as word-of-mouth recommendations used by people daily. Yenta deals currently with text such as electronic mail messages, the contents of a user's files in a directory, etc. For example, two users, A and B, are deemed to share the same interest if A has at least one *granule* of interest as B. A granule may represent the fact that a user reads regularly newsgroups on politics say. Matchmaking presents some challenging problems which are covered in Foner's paper.

Chavez & Maes (1996) describe some preliminary ideas on Kasbah, a classified ads service on the WWW that incorporates interface agents. Kasbah is meant to represent a 'market place' (a web site) where Kasbah agents, acting on behalf of their owners, can filter through the ads and find those that their users may be interested in, and then proceed to negotiate, buy and sell items. Kasbah-like agents may, in the future, render middlemen or brokers redundant.

Last, but by no means least, is the entertainment selection agent which Maes believes has the best potential of all her application domains to be the next "killer application". For example, the Ringo/HOMR system (Shardanand & Maes, 1995; Maes, 1995a) is a personalised recommendation system for music albums and artists, which exploits interface agents. These agents work by social filtering, i.e. a user's agent finds other agents which are correlated, and recommends whatever films their users like to its own user. Hence, like Yenta agents, Ringo's working is similar to a word-of-mouth approach. Maes (1995b) also describes the ALIVE system which is

> "a virtual environment which allows wireless full-body interaction between a human participant and a virtual world which is inhabited by autonomous agents", p. 112.

However, it presents a much more challenging illustration of how autonomous interface agents may be used in entertainment. Essentially, ALIVE demonstrates how agents can form a link between animated characters, based of Artificial Life models, and the entertainment industry.

We hope it is clear that the potential for these interface agents are large. All these demonstrators have been or are being evaluated with users and the results so far are, in the main, quite promising. For example, Ringo has been used by more than 2000 people (Shardanand & Maes, 1995).

In order to emphasise that the distinction between some of these agent types is quite fuzzy, Lashkari *et al.*'s (1994) paper on collaborative interface agents presents a framework for multi-agent collaboration, and discuss results of a demonstrator based on interface agents for electronic mail. This paper emphasises cooperation between agents more than typical interface agents do.

5.2.4 A Brief Critical Review of Interface Agent Systems Work

Less, the reader gets the impression that interface agents research only proceeds in Maes' group at MIT, we must state that this is certainly *not* the case. We have been biased towards them because we (i.e. BT) have ready access to and close links with Pattie Maes and her work and/or demonstrators. Other work on interface agents include Dent *et al.* (1992) and Hermens & Schlimmer (1993). Dent *et al*. (1992), for example, describe a personal learning apprentice agent research done at CMU. This Calendar APprentice agent (CAP), like Kozierok & Maes' (1993) Calendar Agent, assists the user in managing and scheduling its meetings. Their philosophies are essentially the same and the key difference is in their use of learning techniques: Dent *et al*.'s apprentice uses back-propagation neural network and decision tree learning techniques while Calendar Agent (Kozierok & Maes, 1993) uses memory-based and reinforcement learning. Mitchell *et al.* (1994) summarise results from five user-years of experience over which CAP has learned and evolved a set of several thousand rules that model the scheduling preferences of each of its users. These rules could be augmented or edited by users. Hermens & Schlimmer (1993) and Lang (1995) also describe other learning apprentice interface agents. What may differentiate these agents are their performances.

The key criticism of interface agents is that, so far, they tend to function in stand-alone fashions or, at the most, only engage in restricted and task-specific communication with identical peers (Lashkari *et al.*, 1994), which is why the latter authors have begun addressing this issue. This is not necessarily bad but it would be useful to have interface agents being able to negotiate with their peers as do collaborative agents. Furthermore, as Mitchell *et al.* (1994) note

> "...it remains to be demonstrated that knowledge learned by systems like CAP can be used to significantly reduce their users' workload" (p. 90).

But this is the key motivation for having interface agents in the first place (Maes, 1994). Moreover, Wayner & Joch (1995, p. 95) cite Bob Balaban, a systems architect at Lotus Notes, who argues apparently that most people do not need a smart agent which can look over their shoulders, guess their desires and, proactively, take action. He is quoted as saying "I know exactly what I want", arguing he does not need an agent to try to learn from his behaviour. This viewpoint may be dismissed outright by interface agent researchers, but Balaban's point remains - do people want/need interface agents? It appears it is *not* a forgone conclusion that they do. It may just be another working hypothesis!

5.2.5 Interface Agents: Some Challenges

Following on from the last section, some challenges for interface agents include:

- Demonstrating that the knowledge learned with interface agents can truly be used to reduce users' workload, and that users, indeed, want them;
- Carrying out hundreds of experiments using various machine learning techniques (including soft and evolutionary learning techniques) over several domains to determine *which* learning techniques are preferable for *what* domains and *why*;
- Analysing the effect of the various learning mechanisms on the responsiveness of agents;
- Extending interface agents to be able to negotiate with other peer agents;
- Enhance continually the *competence* of interface agents so that their users' *trust* in them build up over time (Maes, 1994). Other issues which Maes notes include guaranteeing the users' privacy and the legal quagmire which may ensue following the fielding of such agents.
- Extending the range of applications of interface agents into other innovative areas such as entertainment, as ALIVE and HOMR are doing.

However, having stated these, there is no denying the fact that interface agents can/will be deployed in real applications in the short term because they are simple, operate in limited domains and do not, in general, require cooperation with other agents.

5.3 Mobile Agents: An Overview

Mobile agents are computational software processes capable of roaming wide area networks (WANs) such as the WWW, interacting with foreign hosts, gathering information on behalf of its owner and coming 'back home' having performed the duties set by its user. These duties may range from a flight reservation to managing a telecommunications network. However, *mobility is neither a necessary nor sufficient condition for agenthood*. Mobile agents are agents because they are autonomous and they cooperate, albeit differently to collaborative agents. For example, they may cooperate or communicate by one agent making the location of some of its internal objects and methods known to other agents. By doing this, an agent exchanges data or information with other agents without necessarily giving all its information away. This is an important point, not least because the public perception of agents (thanks to the popular computing press) is almost synonymous with mobile agents. For example, Peter Wayner's (1995b) agent text (and there are almost no other agent texts

about currently) is titled ‘Agents Unleashed: A Public Domain Look at Agent Technology’, but it is all about mobile agents. Whilst the ‘unleashed’ in the title gives it away, it is rather subtle - so, Wayner could be accused of reinforcing this rather jaundiced view, that agents equals mobile pieces of software.

Another myth to slay is that mobile agents equals Telescript, the current leading mobile agent operating environment invented at General Magic (Mountain View, CA). Through some very clever marketing, General Magic has managed to put mobile agents ‘on the map’ and link their name simultaneously and inextricably to it. But other mobile agent demonstrators or applications not based on Telescript do exist.

5.3.1 Hypothesis, Motivation and Benefits

The key *hypothesis* underlying mobile agents is that agents need *not* be stationary; indeed, the idea is that there are significant benefits to be accrued, in certain applications, by eschewing static agents in favour of their mobile counterparts. These benefits are largely *non-functional*, i.e. we could do without mobile agents, and only have static ones but the costs of such a move are high. For example, consider the scenario borrowed from Wayner (1995b) where the user is required to write a program that would allow her home computer make a flight reservation for her by accessing several airline reservation databases. She lists all her preferences: non-smoking, departure between 7 and 9.30 am from Baltimore, arrival at Austin before noon, no more than one connection, and no changes at Chicago O’Hare. A static single-agent program would need to request for all flights leaving between these times from all the databases, which may total more than 200 and take up many kilobytes. It would also require a list of all the connections and proceed to narrow down the search. Each of these actions involves sifting through plenty of extraneous information which could/would clog up the network. Besides, she is probably paying for this network time.

Consider the alternative. She encapsulates, object-oriented style, her entire program within an agent which consumes probably less than 2K which roams the network of airline reservation systems, arrive safely and queries these databases locally, and returns ultimately to her home computer, with a schedule which she may confirm or refute. This alternative obviates the high communications costs of shifting, possibly, kilobytes of information to her local computer - which presumably she cannot cope with. Hence, mobile agents provide a number of *practical*, though *non-functional*, advantages which escape their static counterparts. So their motivation include the following anticipated benefits.

- Reduced communication costs: there may be a lot of raw information that need to be examined to determine their relevance. Transferring this raw information can be very time-consuming and clog of the networks. Imagine having to transfer many images just to pick out one. It is much more natural to get your agents to “go” to that location, do a local search/pruning and only transfer the chosen compressed image back across the network[2] . It obviates the need for costly network connections between remote computers as required in remote procedure calls (RPC). It provides a much cheaper alternative as we pay increasingly for network bandwidth and time as CompuServe users already do. In the future we would almost certainly be charged by the byte for bandwidth, though others maintain that bandwidth would be free.

[2] This example is due to my colleague, Barry Crabtree.

- Limited local resources: the processing power and storage on the local machine may be very limited (only perhaps for processing and storing the results of a search), thereby necessitating the use of mobile agents.
- Easier coordination: it may be simpler to coordinate a number of remote and independent requests and only collate all the results locally.
- Asynchronous computing: you can 'set off' your mobile agents and do something else and the results will be back in your mailbox, say, at some later time. They may operate when you are not even connected.
- It provides a natural development environment for implementing 'free market' trading services. New services can come and go dynamically and much more flexible services may co-exist with inferior ones, providing more choices for consumers.
- A flexible distributed computing architecture: mobile agents provide a unique distributed computing architecture which functions differently from the static set-ups. It provides for an innovative way of doing distributed computation.
- Lastly, mobile agents represent an opportunity for a radical and attractive rethinking of the design process in general. Following on from the latter, it turns the conventional design process 'on its head', and some truly innovative products should/would emerge out of mobile agent technology.

5.3.2 How Mobile Agents Work: A Brief Telescript View

Telescript is an interpreted object-oriented and remote programming language which allows for the development of distributed applications (see http: URL2). The interpreter and runtime development environment for the Telescript language is called the *Telescript engine* and a given host can support simultaneously multiple Telescript engines. Figure 5 summarises a part view of the Telescript architecture. It shows just one of these Telescript engines integrated onto an operating system via a programming interface called the Telescript application programmer interface (API). The Telescript Development Environment (TDE) can now be downloaded freely from URL2 and it comprises the engine, browser, cloud manager, debugger and associated libraries.

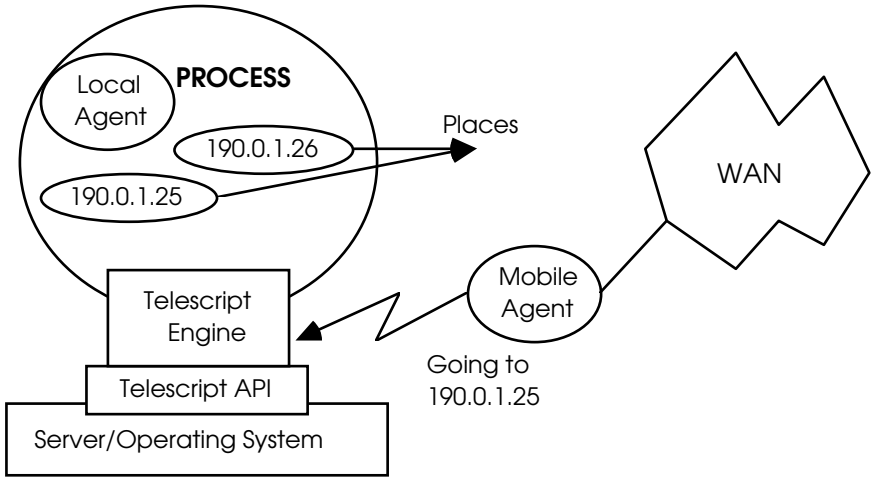

Figure 5 - A Part View of Telescript Architecture (Adapted from Wayner, 1995b)

Telescript applications consist of Telescript agents operating within a 'world' or cyberspace of places, engines, clouds and regions. All of these are objects. For example, a *place* is an instance of some class within the engine whose definition inherits operations which can be called on that place. The top class in Telescript's object hierarchy is the *process*. A Telescript engine is itself a multitasking interpreter which can run multiple processes and switches preemptively between them. Hence, the engine can host multiple agents that share data/information between themselves. Furthermore, a place is itself a process which can contain an arbitrary number and depth of other places. Figure 5 also shows a local agent process. *Agent* processes, unlike *place* processes, are objects which cannot contain other processes, but they can 'go' from place to place (note that places have unique network addresses as shown in Figure 5). An agent requiring a service defined at some given place must go to that place and call the operations there (cf. Figure 5).

So to effect remote programming, Telescript makes use of these three language concepts: *places*, *agents* and *"go"*. "Go" is the primitive which allows for inter-process communication. Two or more agent processes can meet (in one place using the *meet* command) and make use of each other's services. They do this by setting up a communication channel - this is the basis for cooperation. Indeed, by agents moving places, they can exploit the services implemented at these places.

A "go" requires a destination space and the host engine packages up the agent along with all its data, stack and instruction pointer and ships it off to its destination place which may be across a vast WAN. At its destination, the other Telescript-enabled engine unpacks it, checks its authentication, and it is then free to execute at its new place. When it finishes, it returns to its original host having performed the task required by its owner. Non-cooperation occurs when a place refuses to accept an incoming agent process. In the free market model, services would be located at places, and it is up to the agent processes to 'go' there, 'negotiate' for the services, use them, pay and return to their owners.

5.3.3 Mobile Agent Applications

Mobile agent applications do not currently abound but are likely to increase to be in the short to medium term, especially after General Magic's release of their Telescript Development Environment into the public domain with their Open Telescript Initiative. However, the first commercial application was Sony's Magic Link PDA or personal intelligent communicator (PIC) (see http: URL3). Essentially, it assists in managing a user's e-mail, fax, phone and pager as well as linking the user to Telescript-enabled messaging and communication services such as America Online and AT&T PersonaLink Services. The latter, for example, can carry text, graphics and sound. Magic Link operates through the Magic Cap software platform, and a Magic Cap user can send executable agents (Telescript processes) via e-mail through the network. Hence, if two or more users connect their Sony's Magik Link PDAs to AT&T's PersonaLink services (which supports Magic Cap's e-mail messaging), it provides a platform for an application which could exploit email-based Telescript mobile agents.

Plu (1995) mentions that France Telecom, who are a member of the General Magic Alliance and therefore had access to Telescript technology, has prototyped some services based on Telescript. In one of their demonstrators, they have used mobile Telescript agents to integrate railway ticketing and car renting services, and the prototype is able to propose an optimal solution depending on price and time. As noted in Section 3, IBM plans to launch their ICS system which uses mobile agents for providing a communications super-service: capable of

routeing and translating communications from one service and medium to another, e.g. mobile to desktop, PDA to fax, speech to text, etc.

As we write, many others applications are in the pipeline. Telescript technology is now also evolving into active web tool technology (see http: URL3).

5.3.4 A Brief Critical Review of Mobile Agent Systems Work

Telescript is not the only system that permits agents to roam from place to place. In the late 1980s, Siemens developed an application which they called 'Intelligent Moving Processes' (Wolfson *et al*., 1989). In this work, computer programs are interpreted on one machine until a "move" statement is encountered. A 'move' statement causes the packaging of the program, data and instruction pointer (just like with Telescript) and the despatching of this package to a target machine. At the target, a process unpackages the process and the program resumes execution at the new location.

There are other languages which support mobile agent system development notably Java from Sun Microsystems. Java is a programming language similar in syntax to C++, but also similar in other ways to Smalltalk.

It is also important to point out that mobile agent systems need not only be constructed using an agent-oriented system like Telescript. Indeed, Wayner (1995b) shows examples of how mobile agents can be scripted in Xlisp. Other languages to consider include Agent-Tcl, Safe-Tcl and C/C++. Indeed, a couple of years ago at BT Labs, Appleby & Steward (1994) prototyped a mobile agent-based system for controlling telecommunication networks. This system was written completely in C/C++. In this system, there are two types of mobile agents which provide different layers of control in the system. Each node in the network is represented by a routeing table storing the neighbouring node to which traffic should be routed in order for that traffic to reach its particular destination node. The agents control congestion by making alterations to these routeing tables in order to route traffic away from congested nodes. This prototype demonstrated that an ensemble of mobile agents could control congestion in a circuit-switched communications network. In fact, this novel application won the authors a prestigious British Computer Society (BCS) award.

The key criticism of mobile agents is undoubtedly their security. Already, for example, BT Labs operate a 'firewall' which prevents our internal networks being reached from outside. The thought of allowing mobile agents roam into and out of our networks, however benign they are, send shivers up many spines. Telescript agents cannot write to system memory or to disk and so it is safer than viruses which do. However, you can never be too careful as to what roaming agents may leave behind. Furthermore, the range of applications based on mobile agents are rather few, even though this situation will almost certainly have changed by the time this paper is published.

5.3.5 Mobile Agents: Some Challenges

Wayner (1995a) lists the major challenges. They include the following. As usual, they are not exhaustive.

- Transportation: how does an agent move from place to place? How does it pack up and move?

- Authentication: how do you ensure the agent is who it says it is, and that it is representing who it claims to be representing? How do you know it has navigated various networks without being infected by a virus?
- Secrecy: how do you ensure that your agents maintain your privacy? How do ensure someone else does not read your personal agent and execute it for his own gains? How do ensure your agent is not killed and its contents 'core-dumped'?
- Security: how do you protect against viruses? How do you prevent an incoming agent from entering an endless loop and consuming all the CPU cycles?
- Cash: how will the agent pay for services? How do you ensure that it does not run amok and run up an outrageous bill on your behalf?

In addition to these are the following:

- Performance issues: what would be the effect of having hundreds, thousands or millions of such agents on a WAN?
- Interoperability/communication/brokering services: how do you provide brokering/directory type services for locating engines and/or specific services? How do you execute an agent written in one agent language on an agent engine written in another language? How do you publish or subscribe to services, or support broadcasting necessary for some other coordination approaches?

Having listed these, it must be noted that some of them are already being addressed successfully in development environments like TDE using various techniques including the following: using ASCII-encoded, Safe-Tcl scripts or MIME-compatible e-mail messages for transportation; using public-key and private-key digital signature technology for authentication, cash and secrecy; and providing limited and/or interpreted languages that will prevent illegal instructions from being executed, for security; for example, environments would typically not allow an agent to write to memory as viruses do. As a result, much software and hardware (e.g. new consumer electronics products) which exploit mobile agent-based services are currently in the pipeline.

5.4 Information/Internet Agents: An Overview

Information agents have come about because of the sheer demand for tools to help us manage the explosive growth of information we are experiencing currently, and which we will continue to experience henceforth. Information agents perform the role of managing, manipulating or collating information from many distributed sources.

However, before we proceed, perhaps we should clarify that there is, yet again, a rather fine distinction, if any, between information agents and some of those which we have earlier classed as interface or collaborative agents. For example, in Section 5.1.3, we saw the presence of 'information-specific' collaborative agents in the Pleiades distributed architecture (see Figure 3). In Section 5.2.3, we described briefly Sheth & Maes' news filtering agent, NewT, which helps filter and select articles from a continuous stream of Usenet Netnews. We also discussed briefly the Letizia search agent and the remembrance agent. We would not attempt to argue with any researcher who would rather class all these agents as information agents. Interface or collaborative agents started out quite distinct, but with the explosion of the WWW and because of their applicability to this vast WAN, there is now a significant degree of overlap. This is inevitable especially since information or internet agents are

defined using different criteria. They are defined by what *they do*, in contrast to collaborative or interface agents which we defined by what *they are* (i.e. via their attributes, see Figure 1). Many of the interface agents built at the MIT Media Labs, for example, are autonomous and learn, but they have been employed in WWW-based roles; hence, they are, in a sense, information agents. This is a rather subtle distinction, but it must be clarified.

5.4.1 Hypothesis, Motivation and Benefits

> "We are drowning in information but starved of knowledge" (John Naisbitt, Megatrends).

Similarly, vis-à-vis the WWW, it is also the case that we are drowning in data but starved of information. The underlying *hypothesis* of information agents is that, somehow, they can ameliorate, but certainly not eliminate, this specific problem of information overload and the general issue of information management in this information era. We agree with Tom Henry, vice president of SandPoint, who is quoted in Indermaur (1995) as saying that the biggest challenge is to create a simple user interface so that information search and retrieval using information agents will become as natural for people as picking up a phone or reading a newspaper. Though Henry is quoted in Indermaur's article in the context of 'assistant' agents, we believe this is the ultimate *goal* for information agents. Your information agents would, perhaps, put together your own personal newspaper, just as you want it. The information agents would have to be endowed with the capabilities of knowing *where* to look, *how* to find the information and how to collate it.

The case for having information agents should be clearer from the following. Davies & Weeks (1995) report that in 1982, the volume of scientific, corporate and technical information was doubling every 5 years. Three years later, i.e. 1988, it was doubling every 2.2 years, and by 1992 every 1.6 years. This trend suggests that it should now be doubling every year. What is more, much of this information is now accessible electronically on the WWW, whose phenomenal growth over the last 5 years has astonished most. Nicholas Negroponte, head of MIT's Media Labs, claimed in a recent talk at BT Labs that the web was doubling every fifty days. This latter figure is arguable (we believe it is overly optimistic) but the explosive growth of information and the WWW is unquestionable.

The *motivation* for developing information/internet agents is at least twofold. Firstly, there is simply a yearning need/demand for tools to manage such information explosion. Everyone on the WWW would benefit from them in just the same way as they are benefiting from search facilitators such as Spiders, Lycos or Webcrawlers. As Bob Johnson, an analyst at Dataquest Inc., notes:

> "in the future, it [agents] is going to be the only way to search the Internet, because no matter how much better the Internet may be organised, it can't keep pace with the growth in information ...".

Secondly, there are vast financial benefits to be gained. Recall that Netscape Corporation grew from relative obscurity to a billion dollar company almost overnight - and a Netscape or Mosaic client offers generally browsing capabilities, albeit with a few add-ons. Whoever builds the next killer application - the first usable Netscape equivalent of a proactive, dynamic, adaptive and cooperative agent-based WWW information manager - is certain to reap enormous financial rewards. Furthermore, $21 billion was spent by Internet users on purchasing air tickets including hotel bookings, car rentals, etc. in 1995 alone. This compares significantly with the US/European market totals of $170 billion (see http: URL5)

5.4.2 How Information Agents Work

As noted earlier, information agents have varying characteristics: they may be static or mobile; they be non-cooperative or social; and they may or may not learn. Hence, there is no standard mode to their operation.

Internet agents could be mobile, i.e. they may be able to traverse the WWW, gather information and report what they retrieve to a home location. However, this is not the norm as yet. Figure 6 depicts how the typical static ones work. It shows how an information agent, typically within some browser like Netscape, uses a host of internet management tools such as Spiders and search engines in order to gather the information. The information agent may be associated with some particular indexer(s), e.g. a Spider. A Spider is an indexer able to search the WWW, depth-first, and store the topology of the WWW in a database management system (DBMS) and the full index of URLs in the WAIS. Other search/indexing engines or spiders such as Lycos or Webcrawler can be used similarly to build up the index. Indeed, there are currently more than twenty spiders on the WWW.

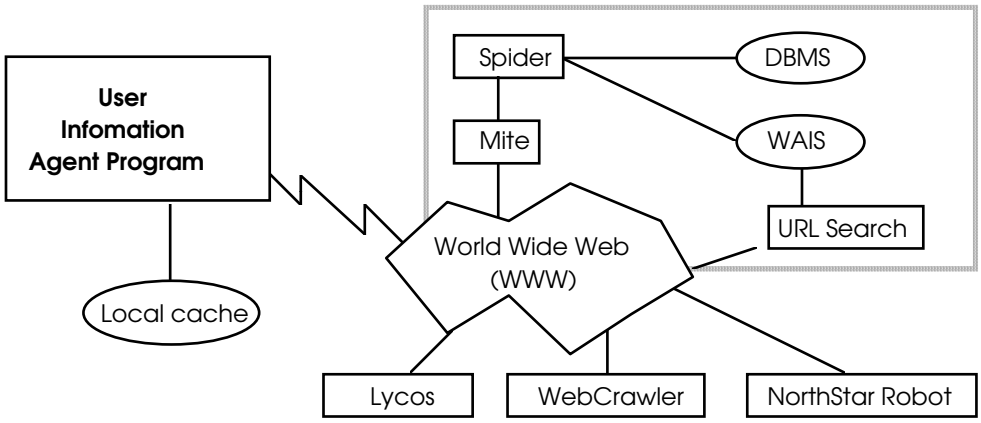

Figure 6 - A view of how Information Agents Work (Adapted from Indermaur, 1995)

The user information agent, which has been requested to collate information on some subject, issues various search requests to one or several URL search engines to meet the request. Some of this search may even be done locally if it has a local cache. The information is collated and sent back to the user.

5.4.3 A Prototypical Example: the Internet Softbot

Etzioni & Weld (1994) describes a state-of-the-art agent called the internet softbot (software robot). It is a fully implemented agent which allows a user to make a high-level request, and the softbot is able to use search and inference knowledge to determine *how* to satisfy the request in the internet. In doing so, it is able to tolerate ambiguity, omissions and the inevitable errors in the user's request. In their paper, Etzioni & Weld use a strong analogy to a real robot in order to describe their softbot-based interface to the internet.. For example, they describe the softbot's *effectors* to include ftp, telnet, mail and numerous file manipulation commands including mv or compress. The *sensors* provide the softbot with information about the external world and they include internet facilities such as archie, gopher and netfind and other Unix commands such as mv or compress; netfind, for example, is used to determine some user's e-mail address.

The contribution of softbot, in its designers' view, is threefold. Firstly, it provides an integrated and expressive interface to the internet. Secondly, it chooses dynamically which facilities to invoke *when* and in *what* sequence. Thirdly, if a UUCP gateway goes down during a search, it is able to backtrack from one facility to another, at run-time in order to try an alternative to meet its goal. This is quite important, not least because the softbot is very goal driven. *Prima facie,* the softbot presents a menu-based interface through which users can compose queries (users are also allowed to use the first-order logic based notation which supports negation, conjunction, quantification and disjunction, but studies have shown that they are uncomfortable with it). However, at its core, the softbot is a goal-driven planner. It translates the filled-in menu form into a softbot goal which it tries to satisfy. It is therefore able to handle tasks such as "send the budget memos to Mitchell at CMU" and "Get all of Ginsberg's technical reports that aren't stored locally". Clearly, there is much disambiguation for the softbot to do, e.g. in the former, who is exactly the intended recipient of the memos? To do this the softbot has to execute a finger mitchell@cmu.edu, *inter alia,* to resolve this. In the latter example, the softbot would need to use the ftp utility, but it would also have to find out where to retrieve Ginsberg's papers, which of his papers are not stored locally (using a combination of universal quantification and negation), and, finally, issue ftp commands to retrieve them. In brief, the planner is the core module which is able to decompose a complex goal expression into simpler ones and go on to solve them. It resolves issues such as interactions between subgoals which it also detects automatically.

Softbots may be implemented for a host of other problems including filtering e-mails, scheduling meetings and performing system maintenance tasks. We classed the softbot as an information agent rather than as an interface agent because learning is not *the* crucial feature of it, though it does some limited memory-based learning. For example, returning to an example given earlier, the softbot is able to record for future reference that it is now familiar with all the Mitchells at CMU - hence, obviating the need to carry out a disambiguation process next time a similar query is received.

5.4.4 A Brief Critical Review of Information Agents Work

We expect information agents to be a major growth area in the next couple of years. At BT Labs, Davies & Weeks (1995) have designed and implemented the Jasper agent - Jasper is an acronym for Joint Access to Stored Pages with Easy Retrieval. Jasper agents work on behalf of a user or a community of users, and is able to store, retrieve, summarise and inform other agents of information useful to them found on the WWW. As a user works with his Jasper agent, a profile of his interests is built dynamically based on keywords. In effect, a Jasper agent is able to 'sit at the side of a user' and suggest interesting WWW pages. Its suggestions are based on a set of keywords given by the user and other 'interesting' WWW pages suggested by other users in the community. These pages are then summarised and keywords are extracted from them which are used to index the pages. If another user's keywords match closely some page, the summary of the page and its URL is e-mailed to the particular user.

There are other information agents built in particular for information filtering. For example, Webwatcher (Amstrong *et al.*, 1995), the RBSE Spider (Eichmann, 1994a) and Metacrawler (http: URL4). The last two are strictly speaking not agents, e.g. Metacrawler is certainly a meta-search engine which provides an interface to other search engines on the WWW. A query submitted to it is translated and forwarded to other search engines; it collates the results and returns them to the user. Spiders are not agents because, even though they explore, autonomously, the topology of the web; generally, they neither learn nor collaborate with other spiders, yet.

The key problem with static information agents is in keeping their indexes up-to-date in an environment which is prone to complete chaos. Some researcher such as Etzioni & Weld (1994) and Eichmann (1994a) have also voiced concerns about the ethics of information agents. We return briefly to such ethical issues towards the end of this paper.

It is probable that the majority of future information agents will be of the mobile variety for similar reasons mentioned in Section 5.3. They would be able to navigate the WWW and store its topology, in a database say, at their home site. The local database may then be queried using SQL.

5.4.5 Information Agents: Some Challenges

This section is much briefer. As regards the challenges of information agents, we believe that they are essentially either similar to those of interface or mobile agents. If the information agents are static, then most of the challenges of interface agents apply (see Section 5.2.5). However, if they are mobile, then most of the challenges for mobile agents are applicable (see section 5.3.4). Likewise, the criticisms of information agents are similar to those of interface and mobile agents depending on whether they are static or mobile respectively.

5.5 Reactive Software Agents: An Overview

Reactive agents represent a special category of agents which do *not* possess internal, symbolic models of their environments; instead they act/respond in a stimulus-response manner to the present state of the environment in which they are embedded. Reactive agents work dates back to research such as Brooks (1986) and Agre & Chapman (1987), but many theories, architectures and languages for these sorts of agents have been developed since. However, a most important point of note with reactive agents are not these (i.e. languages, theories or architectures), but the fact that the agents are relatively simple and they interact with other agents in basic ways. Nevertheless, complex patterns of behaviour *emerge* from these interactions when the ensemble of agents is viewed globally.

Maes (1991a, p. 1) highlights the three key ideas which underpin reactive agents. Firstly, 'emergent functionality' which we have already mentioned, i.e. the dynamics of the interaction leads to the emergent complexity. Hence, there is no *a priori* specification (or plan) of the behaviour of the set-up of reactive agents. Secondly, is that of 'task decomposition': a reactive agent is viewed as a collection of modules which operate autonomously and are responsible for specific tasks (e.g. sensing, motor control, computations, etc.). Communication between the modules is minimised and of quite a low-level nature. No global model exists within any of the agents and, hence, the global behaviour has to emerge. Thirdly, reactive agents *tend* to operate on representations which are close to raw sensor data, in contrast to the high-level symbolic representations that abound in the other types of agents discussed so far.

5.5.1 Hypothesis, Motivation and Benefits

The essential *hypothesis* of reactive agent-based systems is a specification of the *physical grounding hypothesis*, not to be confused with the *physical symbol system hypothesis*. Traditional AI has staked most of its bets on the latter which holds that the necessary and sufficient condition for a physical system to demonstrate intelligent action is that it be a physical symbol system. On the contrary, the physical grounding hypothesis challenges this long-held view arguing it is flawed fundamentally, and that it imposes severe limitations on symbolic AI-based systems. This new hypothesis states that in order to build a system that is

intelligent, it is necessary to have representations grounded in the physical world (Brooks, 1991a). This hypothesis is quite radical and it turns, literally, the physical symbol system hypothesis 'on its head'. Brooks argues that this hypothesis obviates the need for symbolic representations or models because the world becomes its own best model. Furthermore, this model is always kept up-to-date since the system is connected to the world via sensors and/or actuators. Hence, the reactive agents *hypothesis* may be stated as follows: smart agent systems can be developed from simple agents which do not have internal symbolic models, and whose 'smartness' derives from the emergent behaviour of the interactions of the various modules.

It is important to note that all current reactive software agents do not necessarily possess actuators and sensors which connect them to the physical world, though Brooks would insist on them. Indeed, in a paper titled 'Intelligence without Robots', Etzioni (1993) has argued that software environments

> "circumvent many thorny but peripheral research issues that are inescapable in physical environments", p. 7.

However, the essence of the physical grounding hypothesis still holds with such reactive agents: no explicit symbolic representations, no explicit (abstract) symbolic reasoning and an emergent functionality. Reactive agents are simple and easy to understand, and their 'cognitive economy' (Ferber, 1994) is very low; this is because they have to 'remember' little. They are situated, i.e. they do not plan ahead or revise any world models, and their actions depend on what happens at the present moment.

The key *benefits* which motivates reactive agents work, in addition to the hypothesis mentioned earlier, is the hope that they would be more robust and fault tolerant than other agent-based systems, e.g. an agent may be lost but without any catastrophic effects. Other benefits include flexibility and adaptability in contrast to the inflexibility, slow response times and brittleness of classical AI systems. Another benefit, it is hoped, is that this type of work would address the *frame problem* (Pylyshyn, 1987) which has so far proved intractable with traditional AI techniques such as nonmonotonic reasoning.

5.5.2 Reactive Agent Applications

It must be stated that there are a relatively few number of reactive software agent-based applications. Partly, due to this reason, there is no standard mode to their operation; rather, they tend to depend on the reactive agent architecture chosen. We describe briefly two of these architectures below.

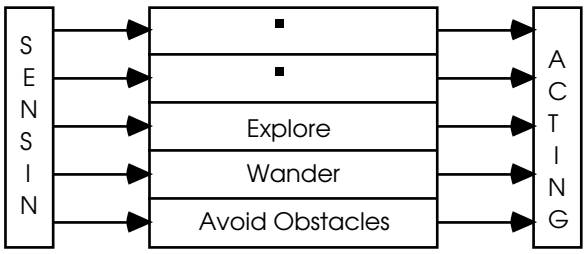

Figure 7 - Brook's Subsumption Architecture

Perhaps the most celebrated of them all, is Brook's (1991) *subsumption architecture*. Though Brook's architecture has been used to implement physical robots (hence tightly connecting

perception to action), it could also be exploited in purely reactive software agents. The architecture consists of a set of modules, each of which is described in a subsumption language based on augmented finite state machines (AFSM). An AFSM is triggered into action if its input signal exceeds some threshold, though this is also dependent on the values of suppression and inhibition signals into the AFSM. Note that AFSMs represent the only processing units in the architecture, i.e. there are no symbols as those in classical AI work. The modules are grouped and placed in layers (which work asynchronously) such that modules in a higher level can inhibit those in lower layers (see Figure 7). Each layer has a hard-wired purpose or *behaviour*, e.g. to avoid obstacles or to enable/control wandering. This architecture has been used to construct, at least, ten mobile robots at MIT. Steels (1990) uses similar agents to Brooks' in order to investigate cooperation between distributed simulated robots using self-organisation.

Arguably, the most basic reactive architecture is that based on situated-action rules which, in turn, derives from some work carried out in by Suchman (1987). Situated action agents act essentially in ways which is 'appropriate' to its situation, where 'situation' refers to a potentially complex combination of internal and external events and states (Connah, 1994). Situated-action 'agents' have been used in PENGI, a video game designed as part of Agre's doctoral thesis (Agre, 1988), and SONJA (Chapman, 1992). Researchers at Philips research laboratories in Redhill, UK, have implemented a situation-action based language called the RTA programming language (Graham & Wavish, 1991). Indeed, they have used this language to implement characters in computer games which they have since integrated into CD-i titles (Wavish & Graham, 1995). Kaebling & Rosenschein (1991) have proposed another language based on a modal logical formalism, which in turn is based on a paradigm called *situated automata*. Agents written in this language are compiled into digital circuits which implement the reactive agent system.

In summary, few applications based on reactive software agents exist currently but this situation will change before the millenium. A favourite application area for them seems to be the games or entertainment industry, which of course is a multi-billion pound industry. For example, the Philips researchers are already working on digital video and 3-D graphics-based, reactive agent animations (Wavish & Graham, 1995).

5.5.3 A Brief Critical Review of Reactive Agents Work

Reactive agent systems can be used to simulate many types of artificial worlds as well as natural phenomena. For example, Ferber (1994) describes how he has used them to simulate ant societies where each ant is modelled as an agent and, a limited ecosystem composed of three kinds of agents: biotapes, shoals of fish and fishermen. As he further explains, reactive agents could make the computer become a "virtual laboratory" where the researcher could modify any experimental parameters and validate his model using both qualitative and quantitative data. Nwana (1993) describes a simulation of children in a playground which was implemented using the Agent Behaviour LanguagE (ABLE), a pre-cursor to RTA (Wavish & Graham, 1994). The ALIVE interactive environment mentioned briefly earlier is is an autonomous system because it employs real sensors in the form of a camera.

Many criticisms can be levelled against reactive software agents and their architectures. Firstly, as already noted, there are too few applications about based on them. Secondly, the scope of their applicability is currently limited, mainly to games and simulations. Even Brooks' robots are yet to deliver useful industrial applications even though we can envisage how they can be exploited in certain applications, e.g. in the toys domain. To be fair, it is still

early days for such research: arguably, symbolic AI did not start delivering any useful industrial applications until the late 1970s or early 1980s, i.e. more than two decades after symbolic AI was born. So, there is a clear need to expand the range of languages, theories, architectures and applications for reactive agent-based systems. Thirdly, it is not obvious how to design such systems so that your intended behaviour emerges from the set-up of agents. How many of such agents are required for some application? Currently, since it is not allowable to tell the agents how to achieve some goal, as with genetic algorithms,

> "one has to find a "dynamics", or interaction loop or servo loop, involving the system and the environment which will converge towards the desired goal. The interaction process only comes to a rest (or a fixed pattern) when the goals are achieved" (Maes, 1991b, 50).

This would not only be time-consuming, but it also smacks of 'trial and error' with all its attendant problems. Fourthly, how are such systems extended, scaled up or debugged? What happens if the 'environment' is changed? Even Brooks (1991a) acknowledges that such questions are frequently asked of his work and so he attempts to tackle them in this paper. However, we do not find his responses very convincing, yet, and perhaps only more applications would improve the trust in the reactive agent hypothesis. Finally, there is the issue of the entire physical grounding hypothesis. Brooks and other *nouvelle AI* researchers argue that the physical symbol system hypothesis

> "implicitly includes a number of largely unfounded great leaps of faith" (Brooks, 1991a, p. 3).

We hope they did not speak too soon: perhaps, the same applies to the physical grounding hypothesis. Etzioni (1993), amongst others, has already challenged Brooks' assertion that the way to make progress in AI is

> "to study intelligence from the bottom up, concentrating on physical systems (e.g. mobile robots), situated in the world, autonomously carrying out tasks of various sorts" (Brooks, 1991c), p. 569.

Furthermore, Maes (1991b) has already pointed out that this situated agents work has some important limitations *precisely* because

> "of their lack of explicit goals and goal-handling capabilities" (p. 50),

requiring the designers of the systems to precompile or hard-wire the action selections. For example, she notes correctly that much effort was expended by the Pengi researchers in analysing the strategies for playing the Pengo game, which were later "hard-wired" into Pengi. Hence, while a planning approach leaves much to the agent, the situated agents approach leaves much to the designers.

Maes (1991b) opted for a more hybrid approach in her agent network architecture. In it she implemented an agent as a set of *competence* modules, each with STRIPS-like (Fikes & Nilsson, 1971) pre- and post- conditions. Modules also get activated if their activation level (a real value) is exceeded, and this level represents the relevance of the module in some situation. If a module has a higher activation level, it will influence the agent's behaviour more. Modules are linked to one another *implicitly* via various links, e.g. a successor link occurs if a module X has a post-condition ß, which happens to be the pre-condition of module Y.

5.5.4 Reactive Agents: Some Challenges

This list of criticisms above is not exhaustive but it provides some of the challenges for reactive agent researchers to address. In summary, we see the main challenges to include the following:

- Expanding the range and number of applications based on reactive agents;
- Methodology: there is a yearning need for a clearer methodology to facilitate the development of reactive software agent applications. This may or may not require the development of more associated theories, architectures and languages. Much of the current approaches, sadly, smacks of 'trial and error';
- Non-functional issues: issues such as scalability and performance would need to be addressed, though these are unlikely to be important until clearer methodologies have been developed and evaluated.

Despite these challenges, we would expect more applications to be 'hand-crafted' in the medium term.

5.6 Hybrid Agents: An Overview

5.6.1 Hypothesis, Motivation and Benefits

So far, we have reviewed five types of agents: collaborative, interface, mobile, internet and reactive agents. The debates as to which of them is 'better' are rather academic, and frankly, sterile - and rather too early to get into. Since each type has (or promises) its own strengths and deficiencies, the trick (as always) is to maximise the strengths and minimise the deficiencies of the most relevant technique for your particular purpose. Frequently, one way of doing this is to adopt a *hybrid* approach, like Maes (1991b), which brought together some of the strengths of both the deliberative and reactive paradigms. Hence, hybrid agents refer to those whose constitution is a combination of two or more agent *philosophies* within a singular agent. These philosophies include a mobile philosophy, an interface agent philosophy, collaborative agent philosophy, etc.

The key *hypothesis* for having hybrid agents or architectures is the belief that, for some application, the benefits accrued from having the combination of philosophies within a singular agent is greater than the gains obtained from the same agent based entirely on a singular philosophy. Otherwise having a hybrid agent or architecture is meaningless. Clearly, the *motivation* is the expectation that this hypothesis would be proved right; the ideal *benefits* would be the set union of the benefits of the individual philosophies in the hybrid. Consider the obvious case of constructing an agent based on both the collaborative (i.e. deliberative) and reactive philosophies. In such a case the reactive component, which would take precedence over the deliberative one, brings about the following benefits: robustness, faster response times and adaptability. The frame problem is also better ameliorated by the reactive component. The deliberative part of the agent would handle the longer term goal-oriented issues. Typically, such hybrid architectures end up having a layered architecture as is evidenced by Muller *et al.*'s (1995) InteRRaP, Ferguson's (1992) Touring Machines, and Hayes-Roth's (1991) architectures. We describe them briefly below.

5.6.2 Hybrid Agent Architectures

As is the case with reactive agents, there are just but a few hybrid agent architectures. A prototypical example of a hybrid example is Muller *et al.*'s layered InteRRaP architecture

shown in Figure 8 developed at the German Research Centre for AI. It is an architecture that implements a layered approach to agent design.

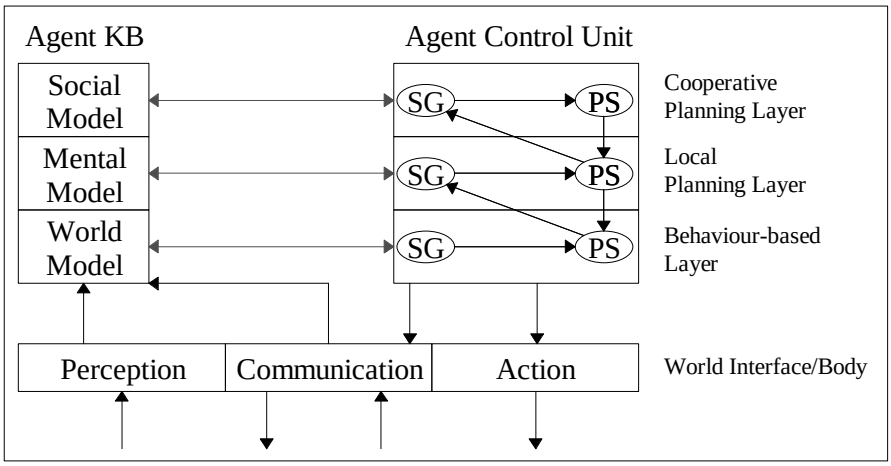

Figure 8 - The InteRRaP Hybrid Architecture (from Fischer *et al*., 1995)

This architecture can be used to construct an agent such as an autonomous robot. As shown, it consists of an agent knowledge base and its associated control unit sitting 'on top' of the perception-action component which also handles the low-level communications. There are three control layers in this architecture: the behaviour-based layer (BBL), the local planning layer (LPL) and the cooperative planning layer (CPL). Clearly, the architecture marries the deliberative and the reactive philosophies. The reactive part of the framework which allows for efficiency, reactivity and robustness are implemented by the BBL which contains a set of patterns of behaviour (PoBs), in effect, situation-action rules. These describe the agents reactive skills which implements fast situation recognition in order to react to time-critical situations. The intermediate LPL implements local goal-directed behaviour while the topmost CPL enables the agent to plan/cooperate with other agents in order to achieve multi-agent plans, as well as resolve conflicts. LPL and CPL allow for more deliberation. These layers all work with different models in the agent's knowledge base: BBL, LPL and CPL operate with the world, mental and social models respectively. Each InteRRaP layer also consists of two processes, SG and PS, which interact with each other as well as with neighbouring layers. These layers work asynchronously. The InteRRaP architecture has been evaluated by constructing a FORKS application which simulates forklift robots working in an automated loading dock environment. For more details on the InteRRaP architecture (whose redesign has been completed recently) and the results of the evaluation, consult Muller *et al.* (1995), Muller (1994) and Fischer *et al.* (1996).

Ferguson's (1992a) *TouringMachines* architecture is another good example of a hybrid "architecture for dynamic, rational and mobile agents" (Ferguson, 1992b), though the word 'mobile' does not refer to mobile agents as in Telescript agents, but to mobile agents as in autonomous robots. This architecture, which is similar to Brook's subsumption architecture (see Figure 7), consists of three control layers: the reactive layer, the planning layer and the modelling layer which all work concurrently. A key distinction between TouringMachines and Brook's subsumption architecture on the one hand, and InteRRaP on the other is that the former are *horizontal* architectures while the latter is a *vertical* architecture. This means that all the layers in TouringMachines and the subsumption architecture have access to the perception/sensing data and all the layers can contribute to the actions (as shown in Figure 7), while only the bottom layer in InteRRaP receives and acts on the perceptual data (see Figure 8). Therefore to achieve coordination in TouringMachines, Ferguson has control rules

capable of suppressing the input to a certain layer, much similar to the suppression/inhibition mechanisms in the subsumption architecture.

Hayes-Roth's (1995) integrated architecture for intelligent agents consists of two layers: the physical layer which performs perception-action coordination, i.e. it senses, interprets, filters and reacts to the dynamic environment in which the agent is embedded; the cognitive layer receives perceptual input from the physical controller to construct an evolving model, and to perform interpretation, reasoning and planning. Her goal is to provide an architecture for constructing adaptive intelligent agents which can operate in specialised, but challenging, "niches"; indeed, she argues cogently that AI agents must, of necessity, be niche-bound because they are knowledge-bound. The fundamental theoretical concept which underlies her architecture captures succinctly the hybridism that belies it: to construct an agent which "dynamically constructs explicit control plans to guide its choices among situated-triggered behaviors", p. 334. Hence, the physical layer implements reactive situated behaviour while the cognitive layer performs some longer term, deliberative planning and scheduling, drawing from the evolving model. Though the Hayes-Roth (1991) paper was largely a design proposal (aiming to provide sophisticated adaptive, intelligent, versatile and coherent agents), Hayes-Roth (1995) reports that the architecture has not only been implemented, but has also been used to implement several experimental agents. For example, she reports on an agent, Guardian, which has been constructed for one niche - Intensive Care Unit (ICU) monitoring. Guardian is currently able to monitor on the order of twenty continuously sensed patient data variables amongst several other occasionally sensed ones. A new demonstrator, Guardian 5, under development will monitor on the order of a hundred variables. She also reports that she has begun applying the architecture to other niches including Aibots - adaptive intelligent robots. Hayes-Roth *et al.* (1995) report on an evolving testbed application - an animated improvisational theatre company for children. The idea is to have animated characters (hybrid agents) which can display spontaneous, situated, opportunistic and goal-directed behaviour. The agents collaborate via *directed improvisation* and the goal is to have the animated characters produce "a joint performance that follows the script and directions in an engaging manner", p. 153.

There are a few other hybrid architectures which we do not review here, an obvious one being the *procedural reasoning system* (PRS) in which the OASIS prototype (Rao & Georgeff, 1995) mentioned in Section 5.1.4 was implemented. The main reference for PRS is Georgeff & Ingrand (1989). Another hybrid system is CIRCA (Musliner *et al*., 1993).

5.6.3 A Brief Critical Review of Hybrid Architectures and Challenges

Hybrid agent architectures are still relatively few in numbers but the case for having them is overwhelming. There are usually three typical criticisms of hybrid architectures in general, not necessarily the ones reviewed above. Firstly, hybridism usually translates to *ad hoc* or unprincipled designs with all its attendant problems. Secondly, many hybrid architectures tend to be very application-specific, and for good reasons too. Thirdly, the theory which underpin hybrid systems is not usually specified. Therefore, we see the challenges for hybrid agents research as quite similar to those identified for reactive agents (see Section 5.5.4). In addition to these, we would also expect to see hybrids of other philosophies than reactive/deliberative ones. For example, there is scope for more hybrids within a singular agent: combining the interface agent and mobile agent philosophies which would enable mobile agents to be able to harness features of typical interface agents or some other combination.

5.7 Heterogeneous Agent Systems: An Overview

Heterogeneous agent systems, unlike hybrid systems described in the preceding section, refers to an integrated set-up of at least two or more agents which belong to two or more different agent classes. A heterogeneous agent system may also contain one or more hybrid agents. As for the other classes, we next discuss their motivation, benefit, how they work, an example and some challenges.

5.7.1 Hypothesis, Motivation and Benefits

Genesereth & Ketchpel (1994) articulate clearly the *motivation* for heterogeneous agent systems. The essential argument is that the world abounds with a rich diversity of software products providing a wide range of services for a similarly wide range of domains. Though these programs work in isolation, there is an increasing demand to have them *interoperate* - hopefully, in such a manner such that they provide 'added-value' as an ensemble than they do individually. The *hypothesis* is that this is plausible. Indeed, a new domain called *agent-based software engineering* has been invented in order to facilitate the interoperation of miscellaneous software agents. A key requirement for interoperation amongst heterogeneous agents is having an agent communication language (ACL) via which the different software 'agents' can communicate with each other. The potential *benefits* for having heterogeneous agent technology are several:

- Standalone applications can be made to provide 'value-added' services by enhancing them in order to participate and interoperate in cooperative heterogeneous set-ups;
- The legacy software problem may be ameliorated because it could obviate the need for costly software rewrites as they be given 'new leases of life' by getting them to interoperate with other systems. At the very least, heterogeneous agent technology may cushion or lessen the blow or effect of routine software maintenance, upgrade or rewrites;
- Agent-based software engineering provides a radical new approach to software design, implementation and maintenance in general, and software interoperability in particular. Its ramifications (e.g. moving from passive modules in traditional software engineering to proactive agent-controlled ones) would only be clear as this methodology and its tools become clearer.

Genesereth & Ketchpel (1994) note that agent-based software engineering is often compared to object-oriented programming in that an agent, like an object, provides a message-based interface to its internal data structures and algorithms. However, they note that there is a key distinction: in object-oriented programming, the meaning of a message may differ from object to object (this is the principle of polymorphism); in agent-based software engineering, agents use a common language with an agent-independent semantics. They highlight three important questions raised by the new agent-oriented software engineering paradigm. They include (p. 48):

- What is an appropriate agent communication language?
- How are agents capable of communicating in this language constructed?
- What communication architectures are conducive to cooperation?

In their paper, they begin addressing such issues via ACL - an agent communication language they have been developing as part of a DARPA initiative. ACL, *inter alia*, consists of the

Knowledge Interchange Format (KIF), the Knowledge Query and Manipulation Language (KQML) (Finin & Wiederhold, 1991) and Ontolingua (Gruber, 1991).

5.7.2 How Heterogeneous Agent Systems Work

To commence, we provide the rather *specific* definition of the word 'agent' proffered in agent-based software engineering. It defines a software agent as such

> "if an only if it communicates correctly in an agent communication language" (Genesereth & Ketchpel, 1994, p. 50).

If new agents are constructed such that they abide by this dictum, then putting them together in a heterogeneous set-up is possible, though not trivial. However, with legacy software, they need to be converted into software agents first. The latter authors note that there are three ways of doing this conversion. Firstly, the legacy software may totally be rewritten to meet the criteria for agenthood - a most costly approach. Secondly, a *transducer* approach may be used. The transducer is a *separate* piece of software which receives messages from other agents and translates them into the legacy software's native communication protocol, and passes the messages into the program. Likewise, it also translates the program's responses into ACL which is sent on to other agents. This is the favoured approach in many situations where the code may be too delicate to tamper with or is unavailable. Lastly, another approach is the *wrapper* technique. In this approach, some code is "injected" into the program in order to allow it communicate in ACL. The wrapper can access directly and modify the program's data structures. This is clearly a more interventionist approach which requires the code to be available, but offers greater efficiency than the transduction approach.

Once the agents are available, there are two possible architectures to choose from: one in which all the agents handle their own coordination or another in which groups of agents can rely on special system programs to achieve coordination. The disadvantage of the former is that the communication overhead does not ensure scalability which is a necessary requirement for the future of agents. As a consequence, the latter federated approach (see Figure 9) is preferred typically.

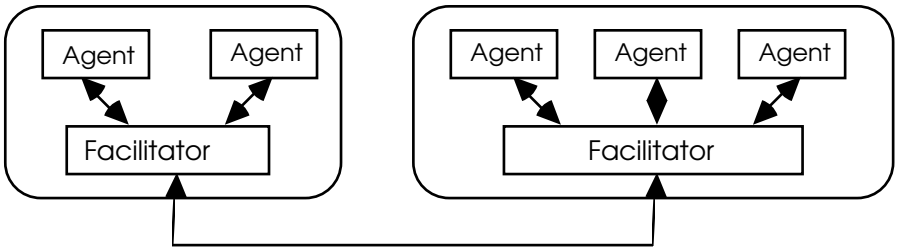

Figure 9 - A Federated System (Adapted from Genesereth & Ketchpel, 1994)

In the above federated set-up, there are five agents distributed in two machines, one with two agents and the other with three. The agents do not communicate directly with one another but do so through intermediaries called facilitators which are similar to Wiederhold's (1992) *mediators*. Essentially, the agents surrender some of their autonomy to the facilitators who are able to locate other agents on the network capable of providing various services. They also establish the connection across the environments and ensure correct 'conversation' amongst agents. ARCHON (Wittig, 1992) used such an architecture.

5.7.3 A Brief Critical Review of Heterogeneous Agent Systems Work

PACT is an acronym for Palo Alto Collaborative Testbed which exemplifies the heterogeneous agents approach. It is an interesting experiment which begins to examine

> "the technological and sociological issues of building large-scale, distributed concurrent engineering systems. The approach has been to integrate existing multi-tool systems that are themselves frameworks, each developed with no anticipation that they would subsequently be integrated" (Cutkosky *et al*., 1993).

The prototype they built integrated four legacy concurrent engineering systems into a common framework. More specifically, it involved thirty one agent-based programs executing on fifteen workstations and microcomputers. The agents were organised into a hierarchy based around facilitators (see Figure 9). Agents communicate with other agents via their facilitators. This PACT experiment was also part of the DARPA knowledge sharing effort.

A related area to heterogeneous agent systems is the new discipline of Intelligent and Cooperative Information Systems (ICIS), born in 1992, which seeks to integrate information systems, software engineering, databases and AI by using *information* agents. Papazoglou *et al.* (1992) describe such a framework which integrates geographically-dispersed database and knowledge base systems (KBSs). In ICIS, an agent is attached to each database or information system, and thus they behave like their front-ends, i.e. a transduction approach is used. This framework allows for requests for some global piece of information that cuts across these databases and KBSs. The requests need to be decomposed by information agents into sub-requests and disseminated to the appropriate systems and the responses are later collated. Another similar architecture is the Carnot architecture (Huhns *et al*., 1993) at MCC which is also addressing the problem of logically unifying physically distributed, enterprise-wide, heterogeneous information. The essential component of Carnot agents are the Extensible Service Switches (ESSs) which are the communication aides to the legacy systems. Essentially, ESSs are facilitators which enable both syntactic and semantic communication between the heterogeneous information systems. Unlike in the agent-based software engineering paradigm, there is a global schema to describe the information in the databases. All communication between two information systems, A and B say, is as follows: the query in A's local context is translated to the global schema which in turn gets translated to B's local context and vice versa.

The work on heterogeneous agent systems is ongoing and there is a need for methodologies, tools, techniques and standards for achieving such interoperability amongst heterogeneous information sources. The challenges, yet to be met, are captured succinctly in the three questions posed by Genesereth & Ketchpel noted in Section 5.7.1. Such work, as is evidenced in their paper, is already underway.

This concludes our panoramic overview of the different classes of agents identified in Section 4.

6 Some General Issues and the Future of Agents

> "Smart agents are here to stay. Once unleashed, technologies do not disappear" (Norman, 1994, p. 71).

We agree fully with these sentiments which is why we believe agent technology is not a passing fad. We have now overviewed a broad range of work which goes under the banner of 'agents'. We have outlined their various promises as well as their challenges. However, apart

from such technical issues, there is also a range of societal (i.e. social) and ethical problems which are looming. Donald Norman writes:

> "Probably all the major software manufacturers are exploring the use of intelligent agents. Myths, promises, and reality are all colliding. But the main difficulties I foresee are social, not technical. How will intelligent agents interact with people and perhaps more importantly, how might people think about agents?" (Norman, 1994, p. 68).

We disagree with Norman as regards the major technical hurdles ahead; as shown in the previous section, there are some extremely demanding technical issues to be resolved in most of the agent classes reviewed. However, we agree that he poses an extremely pertinent question. There are issues which society would have to grapple with through various legislations and they would be very thorny. They include the following:

- Privacy: how do you ensure your agents maintain your much needed privacy when acting on your behalf?
- Responsibility which goes with relinquished authority: when you relinquish some of your responsibility to software agent(s) (as you would do implicitly), be aware of the authority that is being transferred to it/them. How would you like to come back home after a long hard day at work being the proud owner of a used car negotiated and bought for, courtesy of one of your (Kasbah) software agents? How do you ensure the agent does not run amok and run up a huge credit card bill on your behalf?
- Legal issues: following on from the latter, imagine your agent (which you probably bought off-the-shelf and customised) offers some bad advice to other peer agents resulting in liabilities to other people, who is responsible? The company who wrote the agent? You who customised it? Both? We envisage a new raft of legislation would need to be developed in the future to cover software agents.
- Ethical issues: these would also need to be considered. Already, David Eichmann (1994b) is already concerned enough about the ethics of software agents that he has proposed an agent etiquette for information service and user agents as they gather information on the WWW. They include the following:
    - Agents must identify themselves;
    - They must moderate the pace and frequency of their requests to some server;
    - They must limit their searches to appropriate servers;
    - They must share information with others;
    - They must respect the authority placed on them by server operators;
    - An agent's services much be accurate and up-to-date.

  Etzioni & Weld (1994) have proposed others including:

    - Safety - the agent should not destructively alter the world;
    - Tidiness - the agent should leave the world as it found it;
    - Thrift - the agent should limit its consumption of scarce resources;
    - Vigilance - the agent should not allow client actions with unanticipated results.

However, such issues are not that critical immediately, but would be in the medium to long terms. In the short term, we expect some basic agent-based software to be rolled out e.g. some basic interface agents such as mail filtering or calendar scheduling agents. More basic mobile agent services would also be provided in the short term. We can also predict comfortably that many vendors would claim that their products are agent-based even though they most certainly are not. For example, we are already hearing of 'compression agents' and 'system agents' when 'disk compressers' and 'operating systems' would do respectively, and have done in the past. As Guilfoyle (1995) warns:

> "there is a danger, however, that customers may be disappointed by the gap between colourful press reports about smart agents handling half the work, and the reality of 'if-then' rules for message routing", p. 139.

In the medium term (3 to 5 years), some more decent agent applications would be rolled out for most of the classes of agents overviewed in this paper. Perhaps, there would be collaborative agents and/or integrated heterogeneous agent applications doing limited, but *real* workflow or air traffic management or controlling real telecommunication networks say, etc., rather than just simulations. Useful, but limited, interface agents should be available which perform roles including the following: eager assistants, WWW guides, memory aids, entertainment and WWW filters/critics. Indeed, the HOMR/Ringo system is already being marketed as Firefly by Agents Inc. - a Boston-based agents company. Open Sesame (Caglayan *et al.*, 1996) - another interface agent which employs a neural network learning technique, is already on the market from Charles River Analytics (Cambridge, MA). More mobile and information agent applications and languages would soon be rolled out by vendors. We expect reactive and/or hybrid agent technology to start delivering some real everyday industrial applications during this period. Furthermore, during this medium term, we expect the WWW to be commercialised, to some degree, enabling agents of different classes to play a role in paying for services and performing some restricted buying and selling on our behalf, as Kasbah agents propose to do. We would also start seeing a proliferation of specialist agents conferences: agents in the aviation industry, agents in law, etc. We also envisage that the new domain of agent-based software engineering will grow from strength to strength. In the long term (> 7 years), we would expect to see agents which *approximate* true 'smartness' in that they can collaborate and learn, in addition to being autonomous in their settings. They would possess rich negotiation skills and some may demonstrate what may be referred to, arguably, as 'emotions'. We caution against the usage of such words as the latter, not least because the agent literature abounds with such vocabulary which have subtle and complex meanings in the human context (Smith, 1996a). However, it is also at this stage society would need to begin to confront some of the legal and ethical issues which are bound to follow the large scale fielding of agent technology. In the long term too, agents would also provide another design approach to constructing complex pieces of software.

Indermaur (1994) writes:

> "they can't fly yet, but intelligent agents and smart software are beginning to walk" p. 97.

Our view is different: using Indermaur's metaphor, we argue that limited software agents are just to about to 'crawl' out of research laboratories; apart from a few interface and information agents, there is still a very long way, yet, to them 'walking', yet alone to their adolescence.

In summary, we believe, like Greif (1994), that agents can have an enormous effect, but that this will appear in everyday products as an *evolutionary* process, rather than a *revolutionary*

one predicted by much press coverage. For example, the Ovum figures quoted in Guilfoyle's (1995) paper suggest a seven to eight fold increase in the agents market, in Europe and USA alone by the year 2000 (i.e. in four years time) - this suggests a revolution! Greif notes correctly that agents would initially leverage simpler technologies available in most applications (e.g. word processors, spreadsheets or knowledge-based systems). Then agents would gradually be evolved into more complicated applications. We hope the over optimistic claims about agent technology are moderated. It is a sad fact that the only jarring voices in this hymn of confident approbation about agent technology come from those doing the *real* research, and who know what some of the real technical, social and ethical challenges are.

7 Conclusion and a Postscript

This paper has pilfered from a diverse literature in order to overview, in a single paper, the rapidly evolving area of software agents. Only one other paper (Wooldridge & Jennings, 1995a) has attempted a similar extensive review of this area, which they do from a theories, architecture and languages angle. In this paper, we have overviewed the same area from the viewpoint of the clear diversity of agents being investigated in universities and research laboratories worldwide. We hope it provides a useful contribution to understanding this exciting field of software agents. However, we conclude this paper with the following, arguably controversial and/or casual, postscript.

The word 'agent' is currently in vogue in the popular computing press as it is within the artificial intelligence (AI) and computer science communities. It has become a buzzword because it is both a technical concept and a metaphor. However, its rampant use should conjure up the problems faced with other flamboyant titles including 'artificial intelligence' itself: far too ambitious claims precede the real technical work that follows; the subject tends to attract dilettantes who just 'jump on the bandwagon' in order to make a "quick buck" or for some other ulterior motives; and coupled with another contentious word such as 'intelligent' as in 'intelligent agent', *prima facie* at least, it generates unrealistic expectations of the state-of-the-art. These exaggerated claims, over-expectations and the 'oversell' eventually drown out the real (and sometimes excellent) achievements, which never match the hype. In short, much is promised but little is delivered! The aftermath of all this is usually quite predictable: interest in the area soon wanes and the research funding, gradually or rapidly, disappears. Eventually, the dilettantes leave in search for 'greener pastures' and leave the real researchers to pick up the pieces. Is this the fate for agents research - a passing fad? Hopefully not!

There are several things which the serious agent researchers can do. Firstly, they can drop the 'intelligent' in intelligent agents as we have done in the title of this paper: its connotation, and hence expectations, are much less. Secondly, they could attempt to ensure, where possible, that dilettantes do not publish articles on agents in the popular press, at least not until it has been reviewed by someone whose interest in the area is more than superficial. This is sometimes possible because some experts usually get asked to review such articles before they go to press. Thirdly, they (i.e. experts) should not *themselves* engage in overselling the domain and, lastly, they must always be critical of the progress in the area in order to provide a more realistic appraisal of the state-of-the-art. In this overview paper, we have attempted to abide by the principles of this doctrine. It is up to the reader to judge how successful we have been in meeting these principles in this paper.

8 Acknowledgements

We acknowledge Nader Azarmi for encouraging us to write this paper, particularly through his provision of a fraction of the literature referenced in it. David Griffiths also encouraged the writing of this article. Several informal discussions with Barry Crabtree, Mark Wiegand, Paul O'Brien, Robin Smith and Nader Azarmi have also shaped some of the views propounded in it. Barry, Robin, Nader, Divine Ndumu and Brian Odgers have also provided feedback which has improved the quality of this paper. Lastly, we acknowledge the comments of the three anonymous reviewers, though two of them were not so anonymous afterall. The author bears all the responsibility for any misunderstandings and/or errors therein. The views expressed are also those of the author, and not necessarily those of BT plc. This work was funded by BT Laboratories.

9 References

Agre, P. E. (1988), *The Dynamic Structure of Everyday Life*, Ph.D Thesis, Department of Electrical Engineering and Computer Science, MIT.

Agre, P. E. & Chapman, D. (1987), "Pengi: An Implementation of a Theory of Activity", *Proceedings of the 6th National Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, 268-272.

Ambros-Ingerson, J. & Steel, S. (1988), "Integrating Planning, Execution and Monitoring", In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88)*, St Paul, MN, 83-88.

Amstrong, R., Freitag, D., Jopachims, T. & Mitchell, T. (1995), "Webwatcher: A Learning Apprentice for the World Wide Web", In *Proceedings of the Symposium on Information Gathering from Heterogeneous, Distributed Environments*, AAAI Press.

Appleby, S. & Steward, S. (1994), "Mobile Software Agents for Control in Telecommunications Networks", *BT Technological Journal* **12** (2), 104-113, April.

Bates, J. (1994), "The Role of Emotion in Believable Characters", *Communications of the ACM* **37** (7), 122-125.

Bond, A. H. & Gasser, L. (1988), *Readings in Distributed Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann.

Bratman, M. E., Israel, D, J. & Pollack, M. E. (1988), "Plans and Resource-Bounded Practical Reasoning", *Computational Intelligence* **4**, 349-355.

Brooks, R. A. (1986), "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation* **2** (1), 14-23.

Brooks, R. A. (1991a), "Elephants Don't Play Chess", In Maes, P. (ed) (1991), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, London: The MIT press, 3-15.

Brooks, R. A. (1991b), "Intelligence without Representation", *Artificial Intelligence* **47**, 139-159.

Brooks, R. A. (1991c), "Intelligence without Reason", *In Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Menlo Park, CA: Morgan Kaufmann, 569-595.

Caglayan, A., Snorrason, M., Jacoby, J., Mazzu, J. & Jones, R. (1996), "Lessons from Open Sesame! a User Interface Learning Agent", In *Proceedings the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96)*, London, 22-24 April, 61-74.

Carver, N. & Lesser, V. (1995), "The DRESUN Testbed for Researcg in FA/C Distribution Situation Assessment: Extensions to the Model of External Evidence", In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, USA, June, 33-40.

Carver, N., Cvetanovic, Z. & Lesser, V. (1991), "Sophisticated Cooperation in Distributed Problem Solving", in *Proceedings of the 9th National Conference on Artificial Intelligence* **1**, Anaheim, 191-198.

Chaib-draa, B., Moulin, B., Mandiau, R. & Millot, P. (1992), "Trends in Distributed Artificial Intelligence", *Artificial Intelligence Review* **6**, 35-66.

Chapman, D. (1992), *Vision, Instruction and Action*, London: MIT Press.

Chavez, A. & Maes, P. (1996), "Kasbah: An Agent Marketplace for Buying and Selling Goods", In *Proceedings the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96)*, London, 22-24 April, 75-90.

Cohen, P. R. & Levesque, H. J. (1990), "Intention is Choice with Commitment", *Artificial Intelligence* **42**, 213-261.

Connah, D. (1994), "The Design of Interacting Agents for Use in Interfaces", in Brouwer-Janse, D. & Harringdon, T. L. (eds.), *Human-Machine Communication for Educational Systems Design*, *NATO ASI Series*, Series **F**, Computer and Systems Sciences **129**, Heidelberg: Springer Verlag.

Cutkosky, M. R., Engelmore, R. S., Fikes, R. E., Genesereth, M. R., Gruber, T. R., Tenenbaum, J. M. & Weber, J. C. (1993), "PACT: An Experiment in Integrating Concurrent Engineering Systems", *IEEE Computer* **1**, January, 28-37.

Davies, N. J. & Weeks, R. (1995), "Jasper: Communicating Information Agents", in *Proceedings of the 4th International Conference on the World Wide Web*, Boston, USA, December.

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

Decker, K. S. (1995), "Distributed Artificial Intelligence Testbeds", In O'Hare, G. & Jennings, N. (eds.), *Foundations of Distributed Artificial Intelligence*, Chapter 3, London: Wiley, forthcoming.

Decker, K. S. & Lesser, V. R. (1993), "Designing a Family of Coordination Algorithms", in *Proceedings of the 11th National Conference on Artificial Intelligence*, Washington, 217-224.

Dent, L., Boticario, J., McDermott, J., Mitchell, T. & Zabowski, D. A. (1992), "A Personall Learning Apprentice", In *Proceedings of the 10th National Conference on Artificial Intelligence*, San Jose, California, AAAI Press, 96-103.

Doran, J., Carvajal, H., Choo, Y. & Li, Y. (1991), "The MCS Multi-agent Testbed: Developments and Experiments", in Deen, S. (ed.), *Cooperating Knowledge based Systems*, Heidelberg: Springer-Verlag, 240-251.

Durfee, E. H. & Montogomery, T. A. (1989), "MICE: A Flexible Testbed for Intelligent Coordination Experiments", *In Proceedings of the 1989 Distributed Artificial Intelligence Workshop*, 25-40.

Durfee, E. H., Lesser, V. R. & Corkill, D. (1987), "Coherent Cooperation among Communicating Problem Solvers", *IEEE Transactions on Computers* **C-36**(11), 1275-1291.

Eichmann, D. T. (1994a), "The RBSE Spider -- Balancing Effective Search Against Web Load", *In Proceedings of the First International Conference on the World Wide Web*, Geneva, Switzerland, May 25-27, 369-378.

Eichmann, D. T. (1994b), "Ethical Web Agents", *Proceedings of the 2nd WWW Conference*, http://www.ncsa.uiuc.edu/SDG/IT94/

Etzioni, O. (1993), "Intelligence without Robots: A Reply to Brooks", *AI Magazine* **14** (4), 7-13.

Etzioni, O. & Weld, D. (1994), A Softbot-Based Interface to the Internet", *Communications of the ACM* **37** (7), 72-76.

Ferber, J. (1994), "Simulating with Reactive Agents", In Hillebrand, E. & Stender, J. (Eds.), *Many Agent Simulation and Artificial Life*, Amsterdam: IOS Press, 8-28.

Ferguson, I. A. (1992a), "Towards an Architecture for Adaptive, Rational, Mobile Agents", In Werner, E. & Demazeau, Y. (eds.), *Decentralized AI 3: Proceedings of the 3rd European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-91)*, Amsterdam: Elsevier, 249-262.

Ferguson, I. A. (1992b), *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*, PhD Thesis, Computer Laboratory, University of Cambridge, UK.

Fikes, R. E. & Nilsson, N. J. (1971), "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", *Artificial Intelligence* **2**, 189-208.

Finin, T. & Wiederhold, G. (1991), "An Overview of KQML: A Knowledge Query and Manipulation Language", Department of Computer Science, Stanford University.

Fisher, K., Muller, J. P. & Pischel, M. (1996), "Unifying Control in a Layered Agent Architecture", *Technical Report TM-94-05*, German Research Center for AI - (DFKI GmbH).

Foner, L. (1993), "What's an Agent, Anyway? A Sociological Case Study", *Agents Memo 93-01*, MIT Media Lab, Cambridge, MA.

Foner, L. (1996), "A Multi-Agent Referral System for MatchMaking", In *Proceedings the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96)*, London, 22-24 April, 245-262.

Gasser, L. (1991), "Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems", *Artificial Intelligence* **47**, 107-138.

Gasser, L. & Huhns, M. (1989), *Distributed Artificial Intelligence* **2**, San Mateo, CA: Morgan Kaufmann.

Gasser, L., Braganza, C. & Herman, N. (1987), "MACE: A Flexible Testbed fo Distributed AI Research", In Huhns, M. (ed.), *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, London: Pitman, Chapter 5, 119-152.

Gasser, L. , Rosenschein, J. S. & Ephrati, E. (1995), "Introduction to Multi-Agent Systems", *Tutorial A Presented at the 1st International Conference on Multi-Agent Systems*, San Francisco, CA, June.

Georgeff, M. (1996), "Agents with Motivation: Essential Technology for Real World Applications", *The First International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology*, London, UK, 24th April 1996.

Georgeff, M. P. & Ingrand, F. F. (1989), "Decision-Making in an Embedded Reasoning System", *In Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, Menlo Park, CA: Morgan Kaufmann, 972-978.

Genesereth, M. R. & Ketchpel, S. P. (1994), "Software Agents", *Communications of the ACM* **37** (7), 48-53.

Graham, M. & Wavish, P. R. (1991), "Simulating and Implementing Agents and Multiple Agent Systems", *In Proceedings of the European Simulation Multi-Conferencce*, Copenhagen, June.

Greif, I. (1994), "Desktop Agents in Group-Enabled Products", *Communications of the ACM* **37** (7), 100-105.

Gruber, T. "Ontolingua: A Mechanism to Support Portable Ontologies", *KSL-91-66*, Stanford University Knowledge Systems Laboratory.

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

Hayes-Roth, B. (1991), "An Integrated Architecture for Intelligent Agents", *SIGART Bulletin* **2**, (4), 79-81.

Hayes-Roth, B. (1995), "An Architecture for Adpative Intelligent Systems", *Artificial Intelligence* **72** (1-2), 329-365.

Hayes-Roth, B. Brownston, L. & van Gent, R. (1995), "Multiagent Collaboration in Directed Improvisation", In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, USA, June, 148-153.

Hermens, L. & Schlimmer, J. (1993), "A Machine Learning Apprentice for the Completion of Repetitive Forms", In *Proceedings of the 9th IEEE Conference on Artificial Intelligence Applications*, Orlando, Florida: IEEE Press, 164-170.

Hewitt, C. (1977), "Viewing Control Structures as Patterns of Passing Messages", *Artificial Intelligence* **8**(3), 323-364.

Huhns, M. N. & Singh, M. P. (1994), "Distributed Artificial Intelligence for Information Systems", CKBS-94 Tutorial, June 15, University of Keele, UK.

Indermaur, K. (1995), "Baby Steps", *Byte*, March, 97-104.

Huhns, M. N. , Jacobs, N., Ksieyk, T., Shen, W-M, Singh, M. P. & Cannata, P. E. (1993), "Integrating Enterprise Information Models in Carnot", *Proceedings of the International Conference on Intelligent abd Cooperative Information Systems (ICI-CIS*), 32-42.

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

Jennings, N. R., Varga, L. Z., Aarnts, R. P., Fuchs, J. & Skarek, P. (1993), "Transforming StandAlone Expert Systems into a Community of Cooperating Agents", *International Journal of Engineering Applications of Artificial Intelligence* **6**(4), 317-331.

Jennings, N., Corera, J. M., Laresgoiti, L., Mamdani, E., Perriollat, F., Skarek, P. & Varga, L. (1995), "Using ARCHON to Develop Real-World DAI Applications for Electricity Transportation and Particle Accelerator Control", *IEEE Expert*, Special Issue on Real World Applications of DAI systems.

Kaebling, L. P. & Rosenschein, S. J. (1990), "Action and Planning in Embedded Agents, In Maes, P. (ed) (1991), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, London: The MIT press, 35-48.

Kay, A. "User Interface: A Personal View", In Laurel, B. (ed.), *The Art of Himan-Computer Interface Design*, Reading, Mass: Addison-Wesley.

King, J. A. (1995), "Intelligent Agents: Bringing Good Things to Life", *AI Expert*, February, 17-19.

Kozierok, R. & Maes, P. (1993), "A Learning Interface Agent for Scheduling Meetings", *Proceedings of the ACM-SIGCHI International Workshop on Intelligent User Interfaces*, Florida, 81-93.

Lang, K. (1995), "Newsweeder: Learning to Filter Netnews", *In Proceedings of the Machine Learning Conference*.

Lashkari, Y., Metral, M. & Maes, P. (1994), "Collaborative Interface Agents", In *Proceedings of the 12th National Conference on Artificial Intelligence* **1**, Seattle, WA, AAAI Press, 444-449.

Lesser, V. & Corkill, D. (1981), "Functionally Accurate, Cooperative Distributed Systems, *IEEE Transactions on Systems, Man, and Cybernetics* **C-11**(1), 81-96.

Levitt, R., Cohen, P., Kunz, J., Nass, C., Christiansen, T. & Jin, Y. (1994), "The Virtual Design Team: Simulating how Organisational Structure and Communication Tools affect Team Performance", In Carley, K. & Prietula, M. (eds.) *Computational Organisation Theory*, San Francisco: Lawrence Erlbaum.

Lieberman, H. (1995), "Letizia: An Agent that Assists Web Browsing", In *Proceedings of IJCAI 95*, AAAI Press.

Maes, P. (ed) (1991a), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, London: The MIT press.

Maes, P. (1991b), "Situated Agents Can Have Goals", In Maes, P. (ed) (1991a), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, London: The MIT press, 49-70.

Maes, P. (1994), "Agents that Reduce Work and Information Overload", *Communications of the ACM* **37** (7), 31-40.

Maes, P. (1995a), "Intelligent Software", *Scientific American* **273** (3), September.

Maes, P. (1995b), "Artificial Intelligence meets Entertainment: Lifelike Autonomous Agents", *Communications of the ACM* **38** (11), November, 108-114.

Minsky, M. (1985), *The Society of Mind*, New York: Simon & Schuster.

Mitchell, T., Caruana, R., Freitag, D., McDermott, J. & Zabowski, D. (1994), "Experience with a Learning Personal Assistant", *Communications of the ACM* **37** (7), 81-91.

Moukas, A. (1996), "*Amalthaea*: Information Discovery and Filtering using a Multiagent Evolving Ecosystem", *Proceedings the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96)*, London, 22-24 April, 421-436.

Muller, J. P. (1994), "A Conceptual Model for Agent Interaction", In Deen, S. M. (ed.), *Proceedings of the 2nd International Working Conference on Cooperative Knowledge Based Systems (CKBS-94*), Keele University: Dake Centre, 213-234.

Muller, J. P., Pishel, M. & Thiel, M. (1995), "Modelling Reactive Behaviour in Vertically Layered Agent Architectures", In Wooldridge, M. & Jennings, N. (eds.) (1995b), *Intelligent Agents*, Lecture Notes in Artificial Intelligence **890**, Heidelberg: Springer Verlag, 261-276.

Musliner, D. H., Durfee, E. H. & Shin, K. G. (1993), "CIRCA: A Cooperative Intelligent Real-Time Control Architecture", *IEEE Transactions on Systems, Man Cybernetics* **23**.

Newell, A. (1982), "The Knowledge Level", *Artificial Intelligence* **18**, 87-127.

Norman, D. (1994), "How might people interact with agents", *Communications of the ACM* **37** (7), 68-76.

Nwana, H. S. (1993), "Simulating a Children's Playground in ABLE", *Working Report*, Department of Computer Science, Keele University, UK.

Nwana, H. S. (1996), "The Potential Benefits of Software Agent Technology to BT", *Internal Technical Report*, Project NOMADS, Intelligent Systems Research, AA&T, BT Labs, UK.

Nwana, H. S., Lee, L. & Jennings, N. R. (1996), "Coordination in Software Agent Systems", *British Telecommunications Technology Journal* **14** (4), October.

Nwana, H. S. & Wooldridge, M. (1996), "Software Agent Technologies", *British Telecommunications Technology Journal* **14** (4) , October.

O'Brien, P. & Wiegand, M. (1996), "Agents of Change in Business Process Management", *British Telecommunications Technology Journal* **14** (4), October.

Ovum (1994), Ovum Report on 'Intelligent Agents: The New Revolution in Software".

Papazoglou, M. P., Laufman, S. C. & Sellis, T. K. (1992), "An Organisational Framework for Cooperating Intelligent Information Systems", *Intelligent and Cooperative Information Systems 1*(1), 169-202.

Plu, M. (1995), "Software Agents in Telecommunications Network Environments", *UNICOM Seminar on Intelligent Agents and their Business Applications*, 8-9 November, London, 225-243.

Pylyshyn, Z. W. (1987), (ed.), *The Robot's Dilemma: The Frame Problem Problem in Artificial Intelligence*, Norwood, NJ: Ablex.

Rao, A. S. & Georgeff, M. P. (1995), "BDI Agents: From Theory to Practice", In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, USA, June, 312-319.

Reinhardt, A. (1994), "The Network with Smarts", *Byte*, October, 51-64.

Rosenschein, J. S. (1985), *Rational Interaction: Cooperation Among Intelligent Agents*, PhD Thesis, Stanford University.

Rosenschein, J. S. & Zlotkin, G. (1994), *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, Cambridge: MIT Press.

Rhodes, B. J. & Starner, T. (1996), "Remembrance Agent: A Continuously Automated Information Retrieval System", In *Proceedings the First International Conference on the Practical Application of Intelligent Agents and Mullti-Agent Technology (PAAM '96)*, London, 22-24 April, 487-496.

Shardanand, U. & Maes, P. (1995), "Social Information Filtering for Automating "Word of Mouth"", In *Proceedings of CHI-95*, Denver, CO., May.

Sheth, B. & Maes, P. (1993), "Evolving Agents for Personalised Information Filtering", In *Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications.*

Shoham, Y. (1993), "Agent-Oriented Programming", *Artificial Intelligence* **60**(1), 51-92.

Smith, R. G. (1980), "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", *IEEE Transactions on Computers* **C29** (12).

Smith, R. (1996a), "Software Agent Technology", *Proceedings ofThe First International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology*, London, UK, 557-571.

Smith, R. (1996b), Personal Communication.

Steels, L. (1990), "Cooperation between Distributed Agents Through Self-Organisation", In Demazeau, Y. & Muller, J. P. (eds.), *Decentralized AI - Proceedings of the 1st MAAMAW*, Amsterdam: Elsevier, 175-196.

Suchman, L. A. (1987), *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge: Cambridge University Press.

Sycara, K. (1995), "Intelligent Agents and the Information Revolution", *UNICOM Seminar on Intelligent Agents and their Business Applications*, 8-9 November, London, 143-159.

Titmuss, R., Winter, C. S. & Crabtree, B. (1996), "Agents, Mobility & Multimedia Information", *Proceedings the First International Conference on the Practical Application of Intelligent Agents and Mullti-Agent Technology (PAAM '96)*, London, 22-24 April, 693-708.

URL1: http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-5/www/pleiades.html.

URL2: http://www.genmagic.com.
URL3: http://www.sel.sony.com.
URL4: http://www.metacrawler.com.
URL5: http://haas.berkeley.edu/~heilman/agents/
Wavish, P. & Graham, M. (1994), "Roles, Skills and Behaviour", In Wooldridge, M. & Jennings, N. (eds.) (1995b), *Intelligent Agents*, Lecture Notes in Artificial Intelligence **890**, Heidelberg: Springer Verlag, 371-386.
Wavish, P. & Graham, M. (1995), "A Situated Action Approach to Implementing Characters in Computer Games", *Applied AI Journal*, (to appear).
Wayner, P. (1995a), "Free Agents", *Byte*, March, 105-114.
Wayner, P. (1995b), *Agents Unleashed: A Public Domain Look at Agent Technology*, Boston, MA: AP Professional.
Wayner, P. & Joch, A. (1995), "Agents of Change", *Byte*, March 94-95.
Wiederhold, G. (1992), "Mediators in the Architecture of Future Information Systems", *IEEE Computer* **25** (3), 38-49.
Wittig, T. (1992) (ed.) *ARCHON: An Architecture for Multi-Agent Systems*, London: Ellis Horwood.
Wolfson, D., Voorhees,, E. & Flatley, M. (1989), "Intelligent Routers" In *Proceedings of the 9th International Conference on Distributed Computing Systems* **DCS-9**, Newport Beach, CA, June 5-9, IEEE Computer Society Press, 371-376.
Wooldridge, M. (1995), "Conceptualising and Developing Agents", In *Proceedings of the UNICOM Seminar on Agent Software*, 25-26 April, London, 40-54.
Wooldridge, M. & Jennings, N. (1995a), "Intelligent Agents: Theory and Practice", *The Knowledge Engineering Review* **10** (2), 115-152.
Wooldridge, M. & Jennings, N. (eds.) (1995b), *Intelligent Agents*, Lecture Notes in Artificial Intelligence **890**, Heidelberg: Springer Verlag.
Wooldridge, M., Mueller, J. P. & Tambe, M. (1996), *Intelligent Agents II*, Lecture Notes in Artificial Intelligence **1037**, Heidelberg: Springer Verlag.
Zlotkin, G. & Rosenschein, J. S. (1989), "Negotiation and Task Sharing among Autonomous Agents in Cooperative Domains", *Proceedings of the 11th IJCAI*, Detroit, Michigan, 912-917.