

The Intelligent Assistant: An Overview

Behnam Azvine, David Djian, Kwok Ching Tsui and Wayne Wobcke

Intelligent Systems Research Group
BT Labs, Adastral Park, Martlesham Heath
Ipswich IP5 3RE, United Kingdom
{ben.azvine,david.djian,kc.tsui}@bt.com, wobckew@info.bt.co.uk

Abstract. The Intelligent Assistant (IA) is an integrated system of intelligent software agents that helps the user with communication, information and time management. The IA includes specialist assistants for e-mail prioritization and telephone call filtering (communication management), Web search and Yellow Pages^{®1} lookup (information management), and calendar scheduling (time management). Each such assistant is designed to have a model of the user and a learning module for acquiring user preferences. In addition, the IA includes a toolbar providing a graphical interface to the system, a multimodal interface for accepting spoken commands and tracking the user's activity, and a co-ordinator responsible for managing communication from the system to the user and for initiating system activities on the user's behalf. A primary design objective of the IA is that its operation is as transparent as possible, to enable the user to control the system as far as is practicable without incurring a heavy overhead when creating and modifying the system's behaviour. Hence each specialist assistant is designed to represent its user model in a way that is intuitively understandable to non-technical users, and is configured to adaptively modify its user model through time to accommodate the user's changing preferences. However, in contrast to adaptive interface agents built under the behaviour-based paradigm, the assistants in the IA embrace complex AI representations and machine learning techniques to accomplish more sophisticated behaviour.

1 Motivation

Intelligent software assistants have become a major business opportunity in view of the rapid developments in Internet and Network Computing and the need for computers to better “understand” their users in order to customise and prioritise information and communication.

The market for such systems is large and growing. For example, it is predicted that by 2005 there will be 0.5 billion mobile phone users and 1 billion Internet users. Information and communication overload is already a common problem for computer users, so the scale of this problem is only likely to increase in the future. Software assistants that can ease the burden on users by filtering unwanted

¹ Yellow Pages[®] is a registered trade mark of British Telecommunications plc in the United Kingdom.

communication and searching for and delivering the right information at the right time will alleviate these problems for the user, and provide differentiation in a competitive market-place for the business. Software assistants make computers more user friendly by hiding the complexity of the underlying computational processes (as noted by Maes [1]), hence the increased usability of the systems may expand the potential market for these services, making the problem of information overload even more critical. Thus the general motivation for the IA is to address the problems of information and communication overload in a way that is personalized to the preferences of an individual user.

The application of intelligent assistants is basically in any situation where the combination of human and current computer technology is either too slow, too expensive or under strain. The development of ever more complex systems creates a real need for computers that better “understand” their users and which are capable of more “personalized” service.

Internal representations or models of the user, the environment and the world play an important part in the architecture of intelligent systems, Azvine and Wobcke [2]. Real-world problems are typically ill-defined, difficult to model and with large-scale solution spaces, Bonissone [3]. In these cases precise models are impractical, too expensive or non-existent. Therefore there is a need for approximate modelling techniques capable of handling imperfect and sometimes conflicting information. Soft computing (SC) offers a framework for integrating such techniques.

Learning and adaptation are important features of SC techniques. Fuzzy and probabilistic techniques can be used as the basic mechanism for implementing the reflexive and deliberative functions necessary for building intelligent systems. Approximate reasoning is required in the user model where the information from the user is incomplete, uncertain or vague. Neuro-fuzzy techniques provide us with powerful and efficient neural network learning algorithms to learn interpretable fuzzy rules and automatically adjust their parameters, Nauck *et al.* [4].

The IA is designed to aid the user in aspects of communication, information and time management. The system is organized as an integrated collection of software agents that communicate with one another through message passing. The IA includes specialist assistants for telephone call filtering (the Telephone Assistant), e-mail prioritization (the E-mail Assistant), Web search (the Web Assistant), Yellow Pages lookup (the YPA), and agenda management (the Diary Assistant). The IA also has a number of agents that support the specialist assistants: the Toolbar provides a graphical interface to the IA, the Speech Assistant enables the system to process spoken commands, the Co-ordinator manages communication from the system to the user and initiates actions on the user's behalf, the Profile Assistant maintains a generic user model used by the information management assistants, and the Database Assistant maintains information such as contact details for the user's acquaintances.

By a *software agent*, we mean a system that performs tasks on behalf of its user to help the user accomplish goals: the user thus delegates some responsibility for achieving these goals to the agent. This *delegation* sense of agency

contrasts with the definition of an agent based on some notion of *rationality*, under which (usually) a system has, or is ascribed, beliefs, desires and intentions, or (alternatively) is designed to achieve “optimum” performance in some operating environment (see Russell and Norvig [5] for a general description of agents in Artificial Intelligence, and Nwana [6] for an extensive survey of software agents as well as a discussion on the terminology).

The personal assistant metaphor was promoted by Maes [1] as characterizing a class of software agent that learned from the user as he/she performed a repetitive task over a period of time. The idea was that a system could adapt its behaviour to the user’s habits by “looking over the shoulder” of the user, and hence could improve its performance with only minimal intervention. Some early systems were also known as *interface agents* because they typically provided a single point of contact between a user and a single software system such as a mail handler or calendar system. The systems described in Maes [1] used a combination of memory-based reasoning and reinforcement learning, and were thus aligned with the “behaviour-based” approach to AI favoured by Maes [7].

Another influential metaphor in software engineering is the tool metaphor, under which the system is akin to a tool the user can employ to help perform some task. The user should thus feel in control of the system at all times. In contrast to Maes’s personal assistants, two central principles followed by Malone, Lai and Grant [8] are that systems should be semi-formal and radically tailorable. By *semi-formal*, it is meant that the system should blur the boundary between its internal formal representations and a more intuitively understandable “semi-formal” user’s picture of the system’s operation. By *radically tailorable*, it is meant that the user should be able to manipulate the system’s models and thus tailor its behaviour to meet his/her own individual needs. Although the tool metaphor may seem to preclude delegation of a task to a software agent, our view is that delegation is allowable when the user is confident that the agent’s actions will accord with his/her expectations, thus maintaining a sense of user control over the system as well as some autonomy of the agent.

The IA combines both these metaphors to provide a system that is at once under the user’s control (as a tool) but which is also adaptive (adjusting its behaviour over time by observing the user’s actions), c.f. Azvine and Wobcke [2]. The main reasons for adopting this hybrid design approach are as follows:

- simple learning techniques such as those used by interface agents are often slow to learn appropriate behaviour; a way for the user to define acceptable initial behaviour is therefore desirable from the usability point of view as well as to bootstrap the learning (*hence semi-formal representations*);
- the behaviour of the IA assistants is complex and so the potential for the learning to lead to incorrect behaviour is increased: it is important that the user understands and feels in control of the system if he/she is to delegate responsibility to it (*hence radical tailorability*);
- the individual assistants need to learn complex behaviour: simple adaptation techniques are not always applicable (*hence complex AI representations and machine learning approaches*);

- it is important to minimize the amount of user intervention and to accommodate the user’s changing preferences over time (*hence adaptivity*).

Of course, the effectiveness of the IA (and of any personal assistant) depends to a large extent on striking the right balance between the time saved by the system adapting to the user’s habits and preferences to suggest helpful actions and the time lost from user intervention necessary to modify unhelpful behaviour resulting from learning an incorrect model.

The assistants in the IA use a variety of techniques appropriate to the tasks being performed. The Telephone and E-mail Assistants use Bayesian network models of user priorities and preferences. The Telephone Assistant maintains a set of priorities of the user’s acquaintances, and uses these in conjunction with the caller’s telephone number to determine the importance of an incoming call. The E-mail Assistant computes the importance of each incoming message based on its sender, recipients, size and content (the subject field). Both assistants continually adapt their behaviour as the user’s priorities change, and the models are adjustable by the user at any time.

The Web Assistant and the YPA use a profile of the user’s interests (maintained by a Profile Assistant) to augment search queries with the aim of producing more relevant information. The user profile is represented as a hierarchical structure of clusters of keywords, Soltysiak and Crabtree [9], and is computed from recently read e-mails, Web pages, and various other sources. The YPA includes a back-end database computed from the Yellow Pages semi-display advertisements, and a front end consisting of a natural language query interface with a dialogue management component. WordNet is used to help refine queries by searching for synonyms and hypernyms of query terms when an initial query fails. The YPA also exploits the explicit structure of the Yellow Pages (such as classifications and cross references) to improve the search results. The profile is fully available to the user and can be modified at any time.

The Diary Assistant schedules tasks based on preferences for task times and durations supplied by the user using simple natural language terms, such as *morning* and *late afternoon*, which are interpreted using fuzzy functions. There are two modes of search for constructing schedules: *global* search is used when allocating time slots to a number of tasks, while *local* search is used when adding or changing a single task in a schedule. The local search algorithm uses an iterative improvement strategy based on the heuristic of minimizing changes to task sequences. The user model consists of the meaning of these fuzzy terms and the preference functions for different types of task. At present these are fixed, but in future work we intend to investigate techniques for learning these functions.

The organization of the remainder of this chapter is based on the components of the IA. We begin with a discussion of related work on software agents. Then we describe the components of the IA in more detail. We first describe the main user interface (the Toolbar and Multimodal User Interface), then the Coordinator, before focusing on communication management (the Telephone and E-mail Assistants), information management (the Web and YPA Assistants), and time management (the Diary Assistant). We conclude with a simple scenario

illustrating the interaction between the various assistants to create an integrated intelligent personal assistant.

2 Related Work

There have been numerous personal assistants described in the literature; all that we know are designed for helping the user perform a single task. Perhaps the earliest reference to interfaces as agents is the UCEgo question answering component of the Unix Consultant described by Chin [10]. This system attempted to infer the user's goals and answer questions appropriately. Maes [1] describes interface agents for a mail handler and a calendar system. In each case, the system used memory-based reasoning and reinforcement learning to predict the user's response in a very specific situation, such as determining the right folder for storing mail or whether to accept, decline or request renegotiation of a proposed meeting time. Part of the rationale for the IA project is to extend the approach to more difficult tasks which require more sophisticated user models and more complex learning algorithms.

Another strand of related work stems from the learning apprentice metaphor. One learning apprentice system in a domain partially overlapping that of the IA is the CAP system described by Dent *et al.* [11]. This system aimed to learn rules for preferred meeting times, locations, etc, by observing a single user over a period of time. CAP used a machine learning classification algorithm, ID3, although neural networks were also tried. One major problem with such a system is that the learning takes a long time to determine useful rules, and it is assumed that the user's preferences are static throughout this learning period.

A more complex task requiring a system of software agents is the visitor hosting task, in which a schedule of meetings for a visitor to the host's institution is to be found that satisfies the constraints and preferences of the people who match the visitor's interests. An early such system was described by Kautz *et al.* [12]. In this system, user agents communicated via e-mail with a task agent (the visitor bot) to collect preferences, and a standard software package was used to determine the final schedule. A more detailed architecture for a similar application was specified by Sycara and Zeng [13]. They proposed the TCA (Task Control Architecture) in which multiple task agents were connected to multiple interface agents and multiple information agents. In contrast to the system of Kautz *et al.*, this system emphasized the planning and co-ordination requirements of the task, and the aim was to produce a generic architecture applicable to a number of problems. Another application using an agent architecture that emphasized the planning capabilities of individual agents was used for information retrieval from multiple sources, see Oates *et al.* [14]. The main task agent was based on the DECAF architecture of Decker *et al.* [15].

What distinguishes the IA from all these earlier systems is that there is no single task that the IA is performing. The system consists of a number of specialist assistants each of which performs a task for which the user can delegate some responsibility to the agent. Each assistant is designed to have its own user model,

its own learning algorithm, and its own preferred interface. This multiplicity of assistants necessitates a simple interface to the overall system. Thus techniques for multimodal user interfaces such as those developed by Wahlster [16] are used in the IA. The IA includes a module for interpreting speech commands and a vision system used for inferring the user's presence and current activity. A similar type of multimodal interface is integrated with the Open Agent Architecture, see Moran *et al.* [17].

Another need resulting from the multiplicity of assistants is that the communications from different assistants to the user must be co-ordinated, so as to avoid overload from many assistants vying for the attention of the user at the same time. The IA includes a special Co-ordinator which maintains information about the user's current and future activities so that communications from an assistant that cannot be delivered when the agent requests can be scheduled for an appropriate time. In addition, the Co-ordinator can enable the IA to proactively respond to the user's goals (corresponding to tasks in the user's diary) by initiating requests that enlist the services of the specialist assistants, for example to conduct a Web search for information related to a forthcoming presentation. The Co-ordinator uses a rational agent architecture based on the PRS system of Georgeff and Lansky [18], extended to enable scheduling and simple rescheduling of system events. Thus the sense of co-ordination used for the IA is somewhat different from other senses used in the literature, which tend to involve commitments between multiple planning agents, c.f. Jennings [19]. It is closer to the sense of co-ordination in Sycara and Zeng [13], extended to handle multiple autonomous agents (in their visitor hosting system, only one agent, an interface agent, communicates with any one user). We sometimes refer to our notion of co-ordination as orchestration, as the Co-ordinator plays a role analogous to the conductor of an orchestra.

3 User Interface

The Toolbar, shown in Fig. 1, is the primary interface between the user and the IA. Although individual assistants typically have their own interface, the purpose of having a toolbar is to keep screen space usage to a minimum, leaving more space on the screen for other applications. The agents corresponding to the icons in Fig. 1 are the Co-ordinator, the E-mail Assistant, the Telephone Assistant, the Database Assistant, the Web Assistant, the YPA, the Speech Assistant, the Diary Assistant, and the Profile Assistant. The border of the E-mail Assistant icon is flashing, indicating that the assistant has information it wants to convey to the user, e.g. the arrival of high priority mail.

The Toolbar serves four major functions:

- notification of events that happen in an assistant;
- control of windows that belong to the assistants;
- control of the system and virtual clock time (for demonstration purposes);
- provision of an alarm clock service.



Fig. 1. Toolbar Interface to the IA

The Toolbar is sensitive to the movement of the mouse. When the mouse is moved across any button, the name of the assistant the button represents is shown as a bubble, as illustrated in Fig. 1, where the mouse has been moved over the icon for the Database Assistant. Its purpose is to provide an easy reference to the user, especially in the early stages of using the IA when he/she may not be familiar with the icons used to represent the assistants.

A second interface to the IA is the Multimodal User Interface (MMUI). The idea of the multimodal user interface is to exploit multiple channels of communication between the user and the application. Common channels of communication include keyboard, mouse, graphical display and the limited use of sound, such as beeps. The MMUI of the IA also employs vision and speech as well as making use of the keyboard to infer information about the user's activities.

In a distributed multi-agent system such as the IA, the MMUI is seen as a service provider to the other assistants. It is up to the subscribers of these services to decide how the information is interpreted and used. For example, the Telephone Assistant may be set to refuse any phone call if the user is not at his/her desk. However, the Web Assistant will not be interested in this presence information when initiating a search request: this only matters when the assistant wants to contact the user.

Speech services in the IA are run in a client/server mode. The Speech Assistant is run as a server and all IA assistants can run a client that connects to the server and configures a recognizer with its own specific vocabulary and grammar. The system used is the Stap system developed at BT Laboratories, Scahill *et al.* [20]. Multiple Stap speech recognizers are run in parallel. When a speech utterance is received, it is dispatched to all recognizers, which then try to find the best match from their own grammar and vocabulary. A simple voting system is used to select the best result from those recognizers that have returned a meaningful output, i.e. not *silence* or *not recognized*.

Visual input has been used in some other applications such as the Smart Work Manager, described more fully in Tsui and Azvine [21] in this volume. The vision system predicts the user's focus of attention using an artificial neural network. Vision information is used in the IA to predict the user's presence and activity, as also described in Tsui and Azvine [21].

An aspect of the multimodal interface that is also demonstrated in the Smart Work Manager is the fusion of information from different sources, e.g. speech and visual input. The goal is to help resolve the command interpreter resolve references in ambiguous commands such as *phone him* using the context supplied by the vision system.

4 Co-ordinator

The individual assistants in the IA are all autonomous; they each perform an activity to help the user repeatedly achieve goals, and they each have their own special interface to the user. Co-ordination is therefore necessary firstly to avoid overloading the user with information from multiple assistants at the same time—this type of co-ordination essentially involves managing communication from the assistants to the user. A more interesting aspect of co-ordination involves proactive behaviour—this refers to the system itself performing tasks on behalf of the user that require the action of more than one assistant (comprising simple plans of action). An important aspect of both types of co-ordination is the performance of actions at appropriate times. As a simple example, a reminder of a meeting should be given at a time close to the meeting and at a time when the user is likely to be available to receive the reminder. The reminder also has a deadline (the start time of the meeting) after which attempting the action is useless. Thus temporal reasoning is central to achieving co-ordination.

The IA achieves both forms of co-ordination through the action of a special Co-ordinator, responsible for maintaining information about the user's state and future activities, and for scheduling its own future actions and (some of) the activities of the other assistants. To avoid communication overload, the Co-ordinator has a simple user interface, shown in Fig. 2, recording information about whether the user is:

- accepting phone calls;
- accepting e-mails;
- accepting interruptions (from any assistant).

This information is updated automatically using information from the Diary Assistant (each entry in the diary has a flag indicating whether or not the task is interruptible), and is used by the other assistants in determining whether or not to perform an action. For example, the Telephone Assistant uses this information directly when responding to an incoming call, checking the user's state before allowing the phone to ring. The state can also be updated directly from the user interface: if the user sets the state to interruptible, the state switches to accepting phone calls and e-mails automatically (and similarly when setting the state to non-interruptible, the state is set to refusing phone calls and e-mails).

The Co-ordinator is distinguished as the only agent in the IA capable of scheduling future tasks. This part of the Co-ordinator is built on a belief-desire-intention agent architecture which is an extension of the PRS system of Georgeff and Lansky [18]. The beliefs of the Co-ordinator are facts about the user state and the future actions of the user as supplied by the Diary Assistant. The goals of the Co-ordinator correspond to (some of) the user's tasks, also determined using information from the Diary Assistant. For each goal, the Co-ordinator schedules system tasks (creates intentions) using a plan template related to the goal type taken from an existing plan library, and subsequently executes the system tasks at the determined time. For example, a reminder to a meeting is scheduled for a time 5 minutes before the user becomes unreachable from

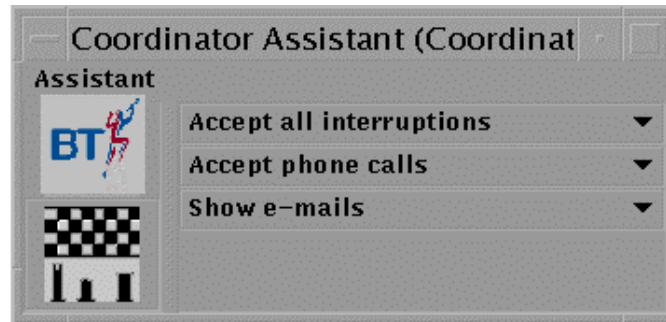


Fig. 2. Co-ordinator Interface

that point until the start of the meeting—the action’s deadline). The user is currently notified of any notifications through the Toolbar: the Co-ordinator’s icon flashes, a beep is emitted, and the message appears in the Co-ordinator’s interface window (Fig. 2). However, other means of communication, such as a mobile phone, could also be used for notification: all that is needed is the technical capability for the IA to initiate phone calls and the use of speech synthesis to generate a voice message.

The Co-ordinator must maintain the timeliness of its scheduled actions. This sometimes means that tasks need to be rescheduled when the user task (goal) time is changed in the diary. For example, when the user changes the time of a meeting, the Co-ordinator must reschedule the reminder to the meeting. In the present implementation, this is done by simply deleting the goal and all its associated tasks (upon notification of the change from the diary), creating a new goal corresponding to the new time and then scheduling new tasks in response to the new goal. Another need for rescheduling occurs when the time comes for an action to be executed which involves notifying the user (such as issuing a reminder) but the user is not reachable for some reason (the Co-ordinator’s model of the user’s tasks is not necessarily complete). In this case, the notification is rescheduled for the next time the user is known to be reachable according to the presently available information (unless this time is past the task’s deadline, in which case the goal is dropped).

The Co-ordinator is also capable of initiating actions to satisfy the user’s implicit goals. These are typically commonly recurring goals such as reading e-mails, which we therefore call *recurrent goals*. To assist in scheduling such actions, the Co-ordinator also has a user model consisting of the preferred times for performing these common actions. The times are expressed using fuzzy terms, so an example fact in such a model is ‘likes to read daily e-mail in late afternoon’. At a particular time each day (presently 11:00 p.m.), the Co-ordinator sets up internal goals corresponding to such tasks, e.g. *read daily e-mails*. However, the action corresponding to this goal is not to schedule a reminder to read daily e-mail, but rather to schedule a time at which the reminder will be scheduled. When determining the time for later scheduling, the user model is consulted so

that (in this case), the scheduling action is performed some time before “late afternoon”. Then at that time, the actual reminder is scheduled, and eventually, the reminder issued. The reason behind this “meta-scheduling” is that when the initial recurrent goal is generated (at 11:00 p.m.), the Co-ordinator is not assumed to have accurate information about the user’s day, so scheduling the reminder then would be useless (or else the time of the reminder would require rescheduling in the event of a conflict). To avoid this, a scheduling event is scheduled for a time when the system can be expected to have more complete information. Full details of the internal agent architecture of the Co-ordinator are given in Wobcke [22] in this volume.

5 Telephone Assistant

The Telephone Assistant (TA) handles incoming telephone calls on behalf of the user, the aim being to minimize disruption caused by frequent calls. For each incoming call, the TA determines whether to interrupt the user (before the phone rings) based on the importance of the caller and on various contextual factors such as the frequency of recent calls from that caller and the presence of a related entry in the diary (e.g. a meeting with the caller). When deciding to interrupt the user, the TA displays a panel indicating that a call has arrived; the user then has the option of accepting or declining to answer the call. The TA uses this feedback to learn an overall priority model for how the user weights the different factors in deciding whether or not to answer a call.

The main difficulty in defining a decision process for the TA is that this mechanism must be:

- *flexible* in order to take into account the uncertainty in the knowledge the system has of the real world, so that a call is *likely* to be urgent, rather than *is* urgent;
- *adaptable* to a change of requirements over time as the result of the user’s priorities changing, for instance when a new project starts or when an old one finishes.

For these reasons, the Telephone and E-mail Assistants (see below) use Bayesian networks to represent both their user models and their internal decision models.

The user model of the TA is a Bayesian network giving a priority for each of the user’s acquaintances; a representation of such a model is shown in Fig. 3. In the figure, the names of callers whose calls are always answered are coloured green, while those who are rejected are coloured red (although note that due to contextual factors, these callers sometimes get through). An important point is that the user does not need to know conditional probability to understand and manipulate the model. All that matters is the intuition that higher priorities mean a greater likelihood of the caller being let through by the TA. Indeed, the actual numbers on the priorities do not matter either; all that matters is that the overall behaviour of the TA accords with the user’s expectations.

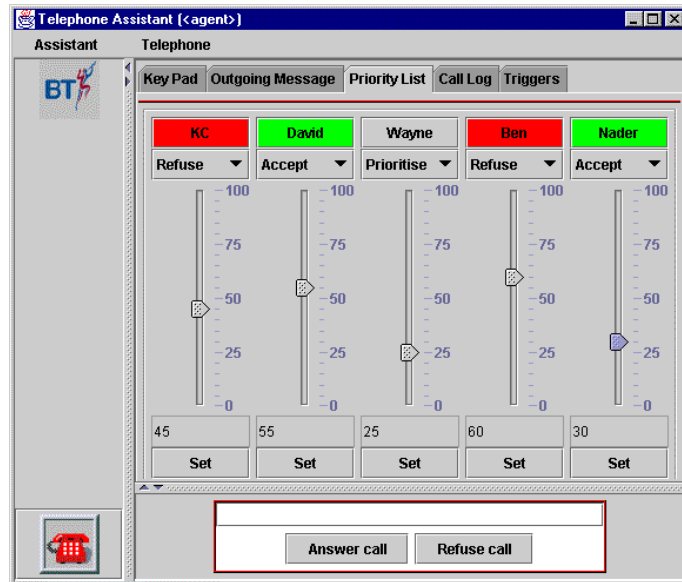


Fig. 3. Telephone Assistant User Model

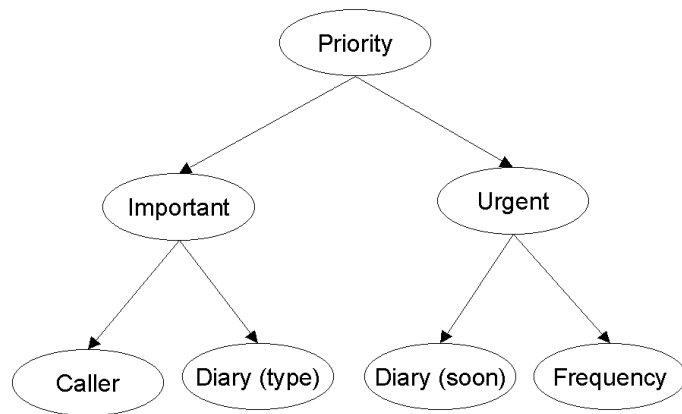


Fig. 4. Telephone Assistant Decision Model

The internal decision model of the TA is also represented as a Bayesian network (not accessible to the user), shown in Fig. 4.

Nodes in the graph correspond to discrete random variables, and arrows represent dependencies used for computation rather than causal links. The TA defines the context of an interruption as represented in the input nodes of the network as follows:

- Caller: the caller’s name as derived from their phone number (obtained using the Database Assistant);
- Frequency: how often this number has called recently (obtained from the TA call log);
- Diary (type): the next type of activity, if any, relating to the caller (obtained from the Diary Assistant);
- Diary (soon): how soon before the activity relating to the caller in Diary (type) will occur (obtained from the Diary Assistant).

Note the distinction between features relating to urgency and features relating to importance. Urgency here refers to the timeliness of a call, importance refers to the usefulness of the information likely to be obtained from a call.

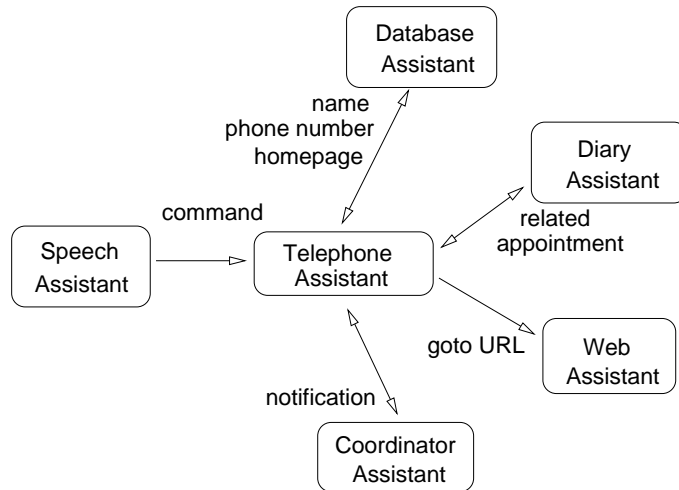


Fig. 5. Telephone Assistant Interactions

The TA makes extensive use of the services of the other assistants. The messages sent and received by the TA are summarized in Fig. 5. Notification messages are received from the Speech Assistant where the message contains a command to execute. Currently, the commands supported are:

- phone Ben;
- phone him.

The Database Assistant is used to identify an incoming caller, and to provide a URL corresponding to the caller's home page; the TA can ask the Web Assistant to display this Web page while the user decides whether to answer a call. As described above, the TA requests information from the Diary Assistant about entries related to the caller in the diary. Finally, the TA requests permission from the Co-ordinator to interrupt the user as a final filter on high priority incoming phone calls. The TA can also initiate a call when the user clicks on a contact's name, using the Database Assistant to perform the translation between name and phone number. The Telephone Assistant is further described in Djian [23] in this volume.

6 E-mail Assistant

The purpose of the E-mail Assistant (EA) is to filter incoming e-mail messages for the user. Based on the sender, recipients, size and content of the message (the subject field), the system determines a suggested time that the message should be read (*now, today, this week, this month, or never*).

As in the Telephone Assistant, the EA uses Bayesian networks to represent its internal decision model. For the sender, the input nodes to the network are similar to the TA caller priority model. For the recipient, the input nodes are whether the message was sent to the user alone, to others or to a distribution list. For size, the input nodes are whether the message is small, medium or large (note that the message could satisfy a number of these conditions to greater or lesser degree). Finally, for the content, the Profile Assistant is used to match the subject field to the user's profile. If there is a match, the priority of the message is increased. This makes the behaviour of the E-mail Assistant tailored to the preferences of the individual user.

The main interface to the E-mail Assistant is shown in Fig. 6. The tabs at the top of the panel indicate the various folders containing already prioritized mail. Current messages are indicated by coloured envelopes at the bottom of the display (red for messages suggested to be read now, orange for today, yellow for this week, white for this month, black for never). Clicking on an envelope results in the corresponding message being displayed, and at this point, the user can enter feedback concerning the appropriate folder in which this message should have been filed. This information is used by the EA to learn the weightings on the various factors contributing to the system's suggestions. The user can also request the EA to file all currently unread mail according to its recommendations.

Fig. 7 shows the messages sent and received by the E-mail Assistant. Currently, the EA only handles notification messages from the Speech Assistant. Speech commands can also be typed in a window accessed from the EA main menu. The speech commands supported by the EA are:

- new mail;
- compose mail.

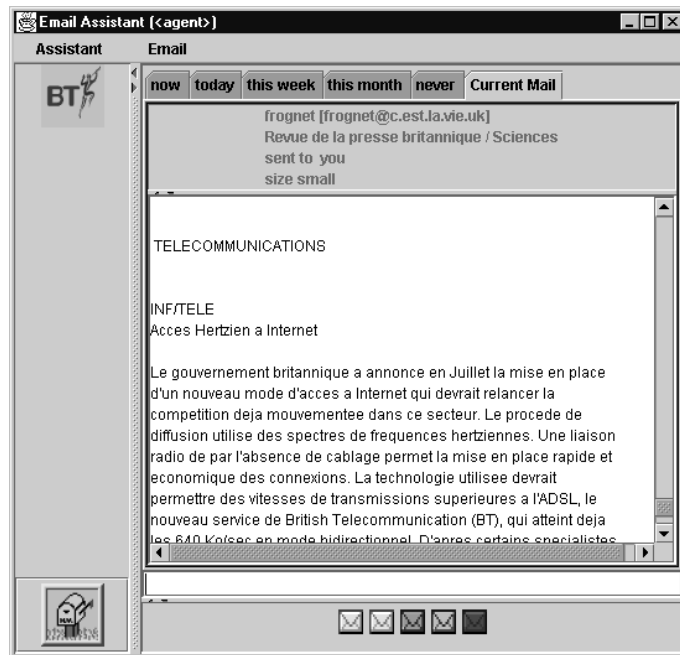


Fig. 6. E-mail Assistant Interface

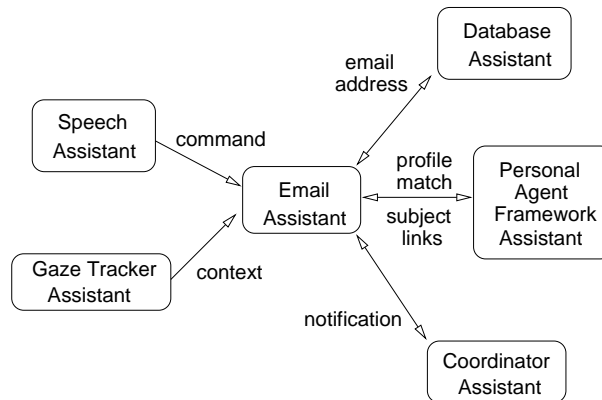


Fig. 7. E-mail Assistant Interactions

The various services that the EA requires from the other assistants are as follows. As for the Telephone Assistant, the Database Assistant provides the contact details of the senders of e-mails. The EA also requests permission from the Co-ordinator to interrupt the user when a high priority (read now) message arrives. When the user composes an outgoing message, the Profile Assistant can also provide hyper-links (URLs) to documents which are relevant to the e-mail. Once the user has typed the subject keywords, the EA sends a query to the Profile Assistant and relevant links are displayed in an applet.

Details of the E-mail Assistant are given in Djian [23] in this volume.

7 Web Assistant

The Web Assistant (WA) provides an interface between the IA and the World Wide Web; it centralizes all the Web search requests in the IA. A component of the WA is a Web browser available as freeware on the Internet that can parse simple HTML files and present results in the same way as commercial Web browsers. This gives the WA the facility to display Web pages in response to requests from other assistants.

When a search request arrives, the WA spawns a thread to deal with the search. As a result, the WA is capable of handling multiple search requests at the same time. The WA is capable of handling search requests initiated both by the user and by the Co-ordinator in response to a user task (in this case, the search keywords are determined by the Co-ordinator from the task description in the diary). The WA then connects to the World Wide Web and formats the result of the search for presentation. It employs a meta-search engine (MetaCrawler) which interacts with several common search engines to perform the actual search. However, before a request is passed to MetaCrawler, the WA tries to augment the keyword(s) using information from the user's profile of interests (obtained from the Profile Assistant) and using an online thesaurus (WordNet).

All search requests are handled as follows:

1. keyword expansion;
2. for each expanded query, contact MetaCrawler to perform search (with 10 second turnaround constraint);
3. combine search results for presentation.

The objective of keyword expansion is to enhance the search to provide more relevant information to the user. Although efficient search engines and meta-search engines are widely available, there are few restrictions on performing multiple searches in terms of turnaround time. The idea is that when a keyword is entered, it is likely that this word represents an area new to the user. However, it is equally likely that the user implicitly wants to find information within an area of his/her current interests. By augmenting the search by keyword expansion, the WA is more likely to satisfy the user's requirements. This is particularly useful as computer users increasingly entrust software agents to look for information on the Internet.

Fig. 8 shows part of the information used by the Web Assistant for keyword expansion: (a) part of the common knowledge, (b) subcategories of the word ‘computer’, and (c) hypernyms of ‘computer’.

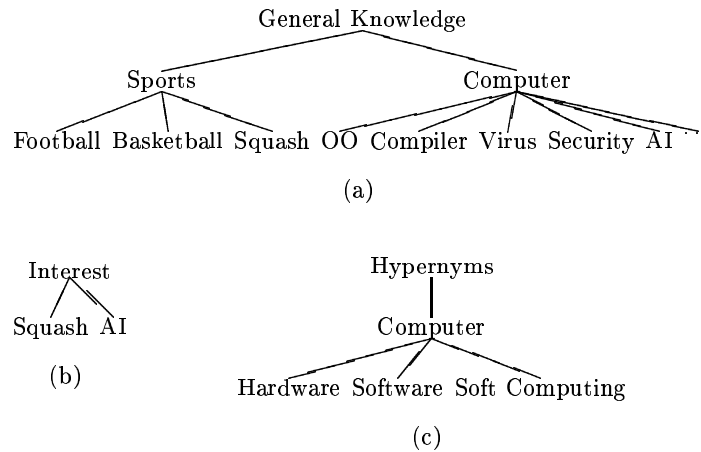


Fig. 8. Information used for Keyword Expansion

For each keyword entered in a request, the hypernym (the more general category to which a word belongs) of the keyword is mapped to the user’s interests. If matching hypernyms are found, new queries are formed in addition to the original query. Then the subcategories for the matched hypernyms are used to form more additional queries. As a result of these two procedures, the number of queries is often increased many times.

As an example, suppose the user asks a query on ‘security’. Based on the user’s interest in Artificial Intelligence, the query is first expanded to ‘computer security’. The subcategories of ‘computer’ in the profile are then used to expand the query to ‘hardware security’, ‘software security’, and ‘Soft Computing security’. All these queries, five in total, are sent to MetaCrawler.

The result page for the above query on ‘security’ is shown in Fig. 9. The first part of the Web page contains the keywords and the expanded keywords, linked to the corresponding anchors containing the search results. For each of the Web pages returned by MetaCrawler, a post-processing method strips out any extraneous information to give a condensed Web page, which is then inserted into the appropriate position in the result page. The user is notified through the Toolbar when the search results are ready.



Fig. 9. Web Assistant Result Page

8 YPA

The YPA provides a natural language query interface to an electronic version of the Yellow Pages [24]. In the IA, the YPA is used to perform personalized search by augmenting queries with the user's interests obtained from the Profile Assistant. Details of the original version of the YPA can be found in De Roeck *et al.* [24] in this volume.

The YPA pre-computes a set of indices to advertisements incorporating classifications, cross references and the free text of the advertisements. The interesting aspect of the Yellow Pages database is that it is *semi-structured*, by which is meant that the entries are a mixture of information indicating structure (such as classifications) and information with no structure (free text).

Within the IA, the YPA is used in conjunction with the user's profile to generate personalized responses. The profile is obtained from the Profile Assistant, and is represented as a hierarchical structure of keyword clusters [9]. Part of an example profile might contain the following:

Food: Italian
Work location: Colchester

Whenever the user requests a restaurant, for example, the YPA assumes that he/she is looking for an Italian restaurant unless another type is specified. A similar rule applies for the location of the restaurant. This aspect of the profile thus provides contextual information for the search with the aim of fleshing out the query to improve the relevance of results. These details can therefore be omitted for convenience when the user enters a query. Additionally, similar to the Web Assistant, the YPA can handle search requests initiated by the Coordinator.

The YPA in the IA has a front end similar to the Web Assistant, shown in Fig. 10 (the interface to the original YPA is much larger). The top text area is for entering any free form text as the search query, while results of the search are shown in the panel below the text area with a tabbed panel for each search result. The original query together with any automatic expansions is shown on the tab. In the example shown, the query ‘24 hour plumber’ is augmented to ‘24 hour plumber in Colchester’, using the location of the user from the profile.



Fig. 10. YPA Interface

9 Diary Assistant

The Diary Assistant (DA) helps the user with time management by maintaining a schedule of tasks. The user is able to enter and edit tasks, move tasks from one day to another, and invoke the scheduling capabilities of the assistant. The main interface to the DA is designed to look like a standard paper diary, with each day divided into half hour slots starting from 9:00 through to 5:00, as shown in Fig. 11. In the display, the duration of each task is given in parentheses beside its description, in the form *hours:minutes*.

The DA uses ideas from fuzzy temporal reasoning, e.g. Dubois and Prade [25]. Users can specify preferences for the start time, duration and deadline of a task using terms that are interpreted as fuzzy functions. At present, the following fuzzy terms are supported: for time preferences *morning*, *afternoon*, *early morning*, *late morning*, *early afternoon*, *late afternoon*, *around t* and *about t* where *t* is a precise time, and when using the global scheduler, *early week*, *mid week*



Fig. 11. Diary Assistant Interface

and *late this week*; and for durations *around d* and *about d* where d is a precise duration. The function denoted by each of these predicates is predefined. As an illustration, fuzzy functions for interpreting various predicates for day periods are shown in Fig. 12. The value of the function at a particular half-hour time indicates the degree to which that time meets the description of the fuzzy term.

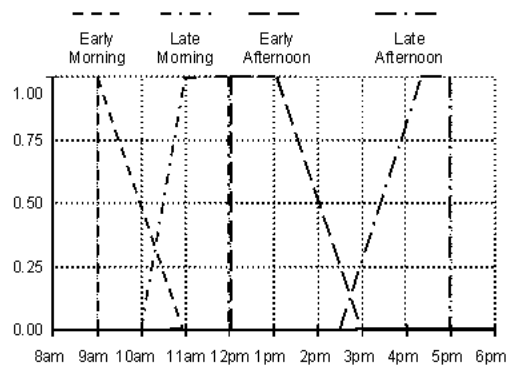


Fig. 12. Sample Fuzzy Functions

The DA incorporates two schedulers: a *global* scheduler for allocating time slots to a set of tasks so as to maximize overall preference satisfaction, and a *local* scheduler that makes minimal adjustments to an existing schedule when adding a single new task. The global scheduler uses a standard AI depth first search algorithm with backtracking, and heuristic guidance to improve the efficiency of the search. The search uses heuristics in two ways: for defining the order in which tasks are considered and the order in which allocations to tasks are explored.

Firstly, tasks are ordered from most constrained to least constrained, where one task is more constrained than another if there are fewer possible time slots that satisfy the user's preference for that task. Secondly, when considering a single task in the context of a partial schedule, the possible time slots are explored in order of preference satisfaction.

The idea behind the use of local scheduling is to minimize disruption to an existing schedule when a *single* new task is added. The need for local search is that the diary entries often involve commitments by the user that should not be changed unless required for the satisfaction of preferences. A "hill-climbing" iterative improvement algorithm is used to implement local search. The procedure starts with a complete task allocation (here including a time slot for the new task), and examines one neighbour of this state (the one with the highest evaluation) to see if this represents a better solution. If it does, local search continues from this state; otherwise the best solution found is returned. The heuristic employed for generating neighbouring states is to consider each legitimate state computed by moving a sequence of tasks of up to length three either forwards or backwards by half an hour. Thus the aim is to minimize changes to *task sequences* in the user's schedules. In the construction of the initial solution which includes the new task added or moved, sequences of up to length three can also be moved forwards or backwards by half an hour, and tasks of imprecise preferred duration can be compressed (possibly in conjunction with moving a preceding or succeeding task half an hour forwards or backwards). This is because sometimes there is no gap in the schedule of sufficient length in which to place the new task; moving or shortening tasks sometimes creates such a gap.

At present, the "user model" of the DA consists of a collection of fixed fuzzy functions, one for each task type, each specifying a value between 0 and 1 for each hour in the week indicating the preference value for tasks of that type to be scheduled at that hour. In future work, we intend to investigate techniques for learning such preference functions. Further details of the Diary Assistant are given in Wobcke [22] in this volume.

10 Integration

One of the main features of the IA is the co-ordinated behaviour produced by integrating the actions of more than one assistant. This occurs through the proactive aspect of the Co-ordinator in responding to some of the user's tasks by initiating plans of actions, some of which invoke the capabilities of the specialist assistants. In this section, we illustrate the complex behaviour underlying the selection, scheduling and execution of a simple plan.

The IA is organized as a collection of autonomous agents communicating using the open messaging architecture of Zeus [26]. Fig. 13 shows (some of) the interactions between the agents in the current version of the IA. An arrow in the figure indicates either the transfer of information or a request for information or to perform a task. Some interactions are a result of the system executing plans in the Co-ordinator plan library.

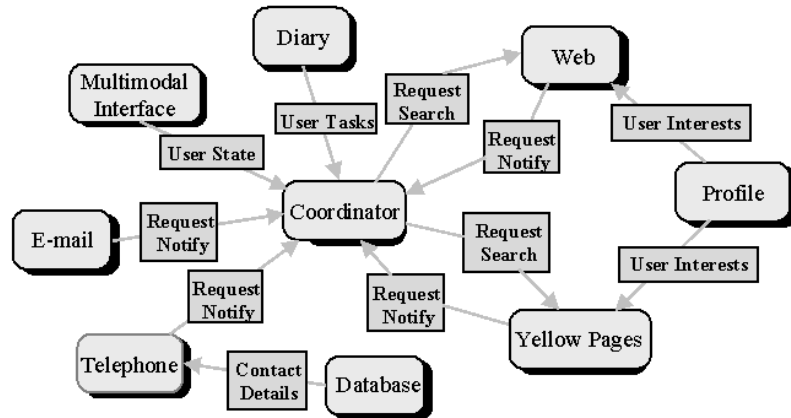


Fig. 13. Agent Interactions

In the IA, tasks in the user's diary, when confirmed by the user, trigger the plans of the Co-ordinator. We describe the simple example of a lunch booking entered in the diary for 1:00 on the current day. The plan related to the lunch booking contains three actions: (i) using the YPA to find a restaurant, (ii) finding the Web page of the person having lunch with the user, and (iii) reminding the user of the lunch appointment. Suppose that the user also enters a non-interruptible 1 hour meeting at 12:00 on the current day.

The whole process starts with the user's confirmation of the diary entries, when messages are sent from the Diary to the Co-ordinator giving the user's latest schedule. The first action of the Co-ordinator is to update its world model on the basis of this information. Then the Co-ordinator treats some of these user tasks, in particular the lunch booking, as goals to be acted upon. For each such goal, the Co-ordinator retrieves the appropriate plan of actions, and creates a new system task for each action in the plan parametrized according to additional information supplied by the user concerning the task (in this case, whom the lunch appointment is with). In this example, the YPA and Web searches are scheduled for the current time, while the reminder is scheduled for 11:55 (5 minutes before the user's non-interruptible meeting), which is 5 minutes before the latest time the user can be assumed to be reachable. The Co-ordinator then executes both the YPA search and Web search request actions, which simply involves sending request messages to the respective assistants.

The YPA augments the query given by the Co-ordinator (to find a restaurant in the user's locale) with the interests of the user (obtained from the user's profile) to search for a restaurant with a particular type of cuisine. The YPA accepts a natural language query such as, in this case, 'Italian restaurant in Ipswich'. The Web Assistant searches for the lunch partner's home page. When the YPA and Web assistants have finished searching, they must notify the user that the results are ready. However, the assistants each request permission from

the Co-ordinator to interrupt the user, again by sending appropriate messages. Should the user be interruptible, permission will be granted by reply; assume otherwise. Then the Co-ordinator will schedule notifications to the user (that the output is ready) for the next available time that the user is interruptible, again based on information supplied from the diary, possibly overridden by the user via the interface. The Co-ordinator performs the notifications and reminders at the determined times.

11 Conclusions and Further Work

The Intelligent Assistant is a working prototype that gives a glimpse of what software agents can do to relieve a user from performing repetitive tasks, and more generally to help them organize work and improve productivity. The individual assistants, specialists in handling e-mail and telephone calls (communication management), Web and Yellow Pages search (information management), and agenda maintenance (time management), are each designed to have an intuitive but powerful user model under the control of the user, and to be adaptive to the user's changing preferences over time so as to minimize the need for direct manipulation. The IA is reactive to the users' commands, but is also capable of proactive behaviour through its planning and co-ordination mechanisms.

The following topics represent research challenges for the various aspects of the Intelligent Assistant.

- *Communication management*: Two issues are the use of more information (richer context) to better model the user's decision making and to learn structures of Bayesian networks representing enhanced priority models.
- *Information management*: One avenue for further work is to incorporate fuzzy query expansion in conjunction with the user profile to handle partial matches more reliably. The personal aspect is important because terms like 'cheap' do not have the same meaning for everyone, nor the same meaning in every context.
- *Time management*: One extension is to provide the Diary Assistant with a negotiation strategy that would enable a number of users to schedule a joint meeting by delegating this task to their agents. For the Co-ordinator, the user could be given more control over the IA's behaviour by allowing the plan library to be dynamically modifiable. Another aspect of the personalization of the IA involves learning plans of user actions. The Diary Assistant could use this information to improve task scheduling and, in commonly occurring situations, to suggest tasks that the user may wish to add to the agenda. The Co-ordinator may be able to suggest plans to be added to the plan library.
- *User Interface*: Further work is needed to develop a method to detect the high level "mood" of the user, a fairly easy task for humans but a challenging one for digital assistants. This would have a major effect on the accuracy of the user models, especially those for communication management, which would have a great impact on the usability of the system.

Acknowledgements

We gratefully acknowledge the contribution to research and development on the Intelligent Assistant made by Simon Case, Gilbert Owusu and Benoit Remael. The YPA is a BT project carried out at the University of Essex by Sam Steel, Udo Kruschwitz and Nick Webb. Thanks also to Nader Azarmi, to Divine Ndumu for help and support with the Zeus environment, and to Barry Crabtree and Stuart Soltysiak for the user profile software.

References

1. Maes P.: 'Agents that Reduce Work and Information Overload', *Communications of the ACM*, Vol. 37, No. 7, pp. 31-40 (1994).
2. Azvine B. and Wobcke W. R.: 'Human-Centred Intelligent Systems and Soft Computing', *BT Technology Journal*, Vol. 16, No. 3, pp. 125-133 (1998).
3. Bonissone P. P.: 'Soft Computing: The Convergence of Emerging Reasoning Technologies', *Soft Computing*, Vol. 1, No. 1, pp. 6-18 (1997).
4. Nauck D., Klawonn F. and Kruse R.: 'Foundations of Neuro-Fuzzy Systems', Wiley, Chichester (1997).
5. Russell S. J. and Norvig P.: 'Artificial Intelligence: A Modern Approach', Prentice-Hall, Englewood Cliffs, NJ (1995).
6. Nwana H. S.: 'Software Agents: An Overview', *The Knowledge Engineering Review*, Vol. 11, No. 3, pp. 205-244 (1996).
7. Maes P.: 'Modeling Adaptive Autonomous Agents', *Artificial Life Journal*, Vol. 1, pp. 135-162 (1994).
8. Malone T. W., Lai K.-Y. and Grant K. R.: 'Agents for Information Sharing and Coordination: A History and Some Reflections', in 'Software Agents', ed., Bradshaw J. M., AAAI Press, Menlo Park, CA (1997).
9. Soltysiak S. J. and Crabtree I. B.: 'Automatic Learning of User Profiles—Towards the Personalisation of Agent Services', *BT Technology Journal*, Vol. 16, No. 3, pp. 110-117 (1998).
10. Chin D. N.: 'Intelligent Interfaces as Agents', in 'Intelligent User Interfaces', eds, Sullivan J. W. and Tyler S. W., ACM Press, New York, NY (1991).
11. Dent L., Boticario J., McDermott J., Mitchell T. M. and Zabowski D. A.: 'A Personal Learning Apprentice', *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 96-103 (1992).
12. Kautz H. A., Selman B., Coen M., Ketchpel S. P. and Ramming C.: 'An Experiment in the Design of Software Agents', *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pp. 438-443 (1994).
13. Sycara K. P. and Zeng D.: 'Coordination of Multiple Intelligent Software Agents', *International Journal of Cooperative Information Systems*, Vol. 5, No. 2, pp. 181-211 (1996).
14. Oates T., Nagendra Prasad M. V., Lesser V. R. and Decker K. S.: 'A Distributed Problem Solving Approach to Cooperative Information Gathering', *Information Gathering from Heterogeneous, Distributed Environments: Papers from the 1995 AAAI Symposium*, pp. 133-137 (1995).
15. Decker K. S., Lesser V. R., Nagendra Prasad M. V. and Wagner T.: 'MACRON: An Architecture for Multi-agent Cooperative Information Gathering', *Proceedings of the CIKM-95 Workshop on Intelligent Information Agents* (1995).

16. Wahlster W.: 'User and Discourse Models for Multimodal Communication', in 'Intelligent User Interfaces', eds, Sullivan J. W. and Tyler S. W., ACM Press, New York, NY (1991).
17. Moran D. B., Cheyer A. J., Julia L. E., Martin D. L. and Park S.: 'Multimodal User Interfaces in the Open Agent Architecture', Proceedings of the 1997 International Conference on Intelligent User Interfaces, pp. 61-68 (1997).
18. Georgeff M. P. and Lansky A. L.: 'Reactive Reasoning and Planning', Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87), pp. 677-682 (1987).
19. Jennings N. R.: 'Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems', The Knowledge Engineering Review, Vol. 8, No. 3, pp. 223-250 (1993).
20. Scahill F., Talintyre J. E., Johnson S. H., Bass A. E., Lear J. A., Franklin D. J. and Lee P. R.: 'Speech Recognition—Making It Work for Real', BT Technology Journal, Vol. 14, No. 1, pp. 151-164 (1996).
21. Tsui K. C. and Azvine B.: 'Intelligent Multimodal User Interface', in this volume.
22. Wobcke W. R.: 'Time Management in the Intelligent Assistant', in this volume.
23. Djian D. P.: 'Communication Management: E-mail and Telephone Assistants', in this volume.
24. De Roeck A., Kruschwitz U., Scott P., Steel S., Turner R. and Webb N.: 'The YPA—An Assistant for Classified Directory Enquiries', in this volume.
25. Dubois D. and Prade H. M.: 'Processing Fuzzy Temporal Knowledge', IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, pp. 729-744 (1989).
26. Nwana H. S., Ndumu D. T. and Lee L. C.: 'ZEUS: An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems', Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pp. 377-391 (1998).