

The Info Agent: an Interface for Supporting Users in Intelligent Retrieval

Daniela D'Aloisi and Vittorio Giannini

Fondazione Ugo Bordoni
Via B. Castiglione 59, I-00142, Rome, Italy
Voice +39 6 5480 3422/3425
Fax +39 6 5480 4405
{dany, gvittori}@fub.it

Abstract

In this paper we present a system that supports users in retrieving data in distributed and heterogeneous archives and repositories. The architecture is based on the metaphor of the software agents and incorporates innovative hints from other fields: distributed architectures, relevance feedback and active interfaces. The system has a cooperative and supportive role: it understands the user's needs and learns from his behavior. Its aim is to disengage the user from learning complex tools and from performing tedious and repetitive actions.

1 Introduction

The storage and retrieval of data has undergone substantial modifications during the years. More recently the development of Internet has turned the classical view of database as a centralized collection of homogeneous data into distributed, heterogeneous collections of information. A datum is not any more a structured entity, but it could be almost any kind of representable item, e.g., a picture, a sound, a text, a record, etc. The spreading of computer networks has made this information available to anyone, and at the same time anyone can play the roles of supply user or demand user [2]. As a consequence there exist a lot of repositories that one could inspect, although only a restrict number of them contain information relevant to a user. In order to face with the multiplicity of data formats, several and different access methods have been developed, although generally addressed to specialists. So a novice or non-expert user has to learn one or more languages to access a different information repositories. There are some basic protocols on which more

sophisticated languages are built. These languages are part of tools for navigating the network that try to take more the user's needs into account, although they do not solve some basic problems:

- The search is based on keywords. Since the user could be imprecise, that is the keywords could not exactly reflect what he is looking for, the set of retrieved documents could be too large or not focussed on the target. Then, the user should repeatedly refine his query to find a satisfactory set of documents. This process is long and could discourage the user in performing further searches.
- The unstructured storage of data makes the search complex since the user could get lost in the information space. Moreover, during the search, the user could be obliged to switch from a language to another while he moves from a site to another.
- The user is not aware of the quantity and type of available information, and he does not know where it is localized. Moreover he cannot be constantly informed about new entries and/or changes in the repositories.

The solution is to provide mechanized supports that help the user in both accessing the information and fully exploiting the resources at his disposal: these supports have to be incorporated into an interface system to assist the users in their needs. A methodology useful to build this type of interface is that of software agent [5]. A software agent dispatches tasks on the user's behalf and according to his directions. The user interacts with the agents without being aware of all the functions performed by them. In particular, an intelligent software agent is able to understand the user's needs by observing his behavior and by sharing sources of knowledge with him.

Our proposal is to design a system that integrates the methodology of the software agents with the Web environment: the aim is to build an intelligent interface that should make a user free of learning complex languages and allows some functions to be automatically performed. The specific points of our proposal are the following:

- Each job is performed by an agent: there can exist simple and complex agents according to their job. Each job can consist of several tasks that are its *atomic* constituents.
- There exists a sort of *Personal Agent* that coordinates all the agents that form the user's support system.
- The agent environment has to be integrated with the Web world. The agents communicate with each other by KQML (Knowledge and Query Manipulation Language) that will have a bridge towards HTML (Hyper Text Markup Language).
- The architecture of the agents must be based on the DAI (Distributed AI) paradigm to allow for incremental and flexible design.

- The support system must be able to understand what the user wants according to the current state of affairs and the user's past history.

In this paper we propose an interface, called the *Info Agent*, based on the previous ideas, that supports the user in retrieving data in remote and local databases and repositories. Moreover it autonomously navigates the network to search documents and data that could interest the user.

In Section 2 the architecture of the *Info Agent* is presented; then in Section 3 the functions offered by the interface are explained. In Sections 4, 5 and 6 the agents forming the interface are described. Section 7 contains some conclusive remarks.

2 An Intelligent Interface for Accessing Distributed Resources

The software agents are intelligent entities that help a user in performing different tasks and supporting him at different stages of his activity (scheduling, searching, communicating, etc.). We have designed and developed a class of information agents (classified in several subclasses) devoted to support different types of users in gathering information and data in local and remote databases through an interface integrated with Netscape. These agents are integrated and presented to the user as a single agent, the *Info Agent*. There are several proposals in literature that describe agents for navigating the Internet [2, 6, 12, 10]. Our architecture presents the following relevant features that characterize it:

- **Distributedness:** a complex task is shared by several agents that cooperate to accomplish it. That happens in a user-transparent manner.
- **Standard and transparent communication:** the agents exchange messages by using the KQML protocol [7].
- **Activeness:** the agents exhibit an active and cooperative behavior since they try to understand the user's needs starting from the past and present states of affairs also when there is no explicit user's request.
- **Cooperation:** the agents use the user's behavior to modify their actions and to decide what to do next.
- **Scalability:** new agents able to perform different types of functionality can be incrementally added to the system.

The whole system consists of several agents specialized in dealing with different type of tasks. An *Interface Agent*, e.g., a sort of Personal Assistant, is in charge of handling with

the user's needs and of connecting him with the agent(s) able to solve his problem(s). The number and the types of agents the *Interface Agent* deals with depends on the aims of the system. Given our architectural choices, e.g., agent-based and distributed architecture, the whole structure can be updated only modifying the *Interface Agent*.

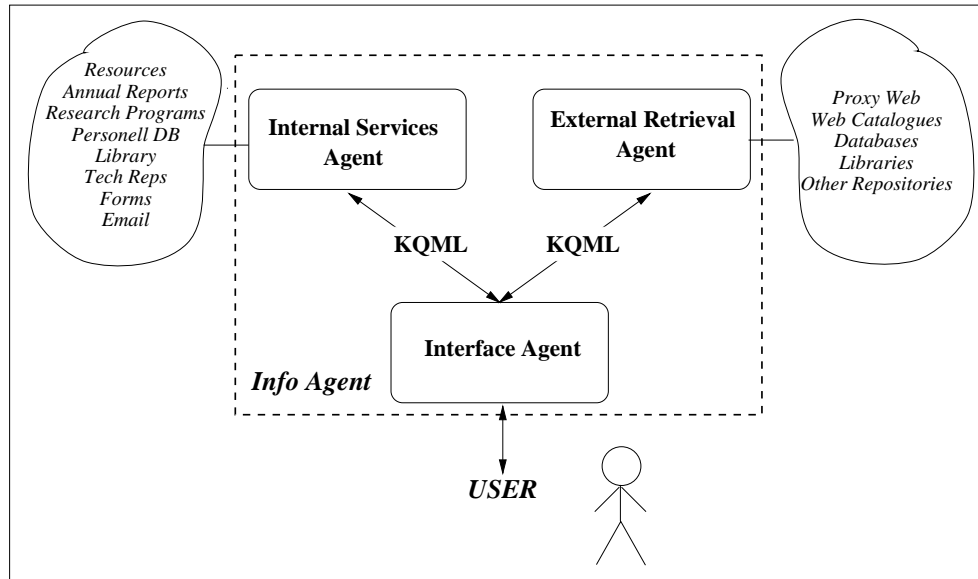


Figure 1: The system architecture

The part of the system concerning the information retrieval agents is depicted in Figure 1 and includes three kinds of agents, the *Interface Agent*, the *External Retrieval Agent* and the *Internal Services Agent*.

The *Interface Agent* interacts directly with the user. Currently the interface is based on the HTML (Hyper Text Markup Language) language, but it is also possible to choose a different type of dialogue (menu, natural language, etc.). Our current choice depends on both the portability of the HTML language and the easy integration in all the Web resources. Moreover, although the selection of the method also depends on the user features, we believe that the chosen front-end is suitable for a large number of users while other methods could result too tedious or too complex. For example, the use of natural language could be boring for an expert and difficult for a novice who does not know the exact terminology.

The *Interface Agent* is able to reason about the user's requests and to understand what type of need he is expressing: it singles out which of the two agents is able to solve the current problem and sends to it its interpretation of the query. The dialogue between the agents occurs in KQML (Knowledge Query and Manipulation Language) [7].

The *Internal Services Agent* knows the structure of the archives available in a given organization: it is in charge of retrieving scientific and administrative data, performing some classes of actions and supporting the user in compiling internal forms. The *External Retrieval Agent* is in charge of retrieving documents on the network. It can work in two modalities: retrieval (or query) mode and surfing mode. In the first case, it searches for a specific document following a query asked by the user: this service is activated by a direct user's request. In the second case, the agent navigates the network searching for documents that, in its opinion, could interest the user. The search is driven by a user's profile built and maintained by the *Interface Agent*. At the beginning, the *Interface Agent* uses a setting introduced by the user; then this profile is refined according to how the user manages the data that the agent finds for and/or proposes him. This agent works autonomously even if the user can deactivate it when he decides to.

Both the agents—the *External Retrieval Agent* and the *Internal Services Agent*—utilize the same software tool to perform their search: it is a public-domain software called Harvest, “an integrated set of tools to gather, extract, organize, search, cache and replicate relevant information across the Internet” [8]. Nevertheless it is also possible to provide other search methods or systems to be used alone or along with Harvest: that is an advantage due to the modular and distributed architecture of the whole framework.

3 Agents and Functions

The implementation of the agents is based on a general model that separates the different high-level functions an agent exhibits. There are three different components each devoted to different classes of performance: communicating, controlling and performing. The communication part is devoted to interact with the user and the external world. The controlling part is devoted to maintain the knowledge components of the agent: it is also in charge of providing reasoning and problem solving competence. The performing part actually executes the actions that are part of the agent's specialization. The functions common to all the agents are part of the basic model that is then specialized according to the specific tasks the agent has to perform. In this paper we do not describe the architecture of the agents: the interested reader can see [3] and [4] for further details.

As said above, the user interacts only with the *Info Agent* that, after having interpreted the user's need, connects him with the agent in charge of solving his problem coordinating the work of different agents when necessary. It is also in charge of defining the information needs of the user by observing his behavior and deducing what type of help or support he needs. The connection with other agents is user-transparent. Currently the *Interface Agent* coordinates two other agents, but the number can increase incrementally in a straightforward manner.

At the present, the user has at disposal the following possibilities:

- *Specific retrieval.*
The user fills in a mask with the *Interface Agent's* help. The query becomes the body of a KQML message that the *Interface Agent* sends to the *External Retrieval Agent* with further references about the site(s) in which documents satisfying the user's request can be retrieved.
- *Surfing.*
The *Interface Agent* activates the *External Retrieval Agent* whose job is to navigate the network searching for documents that could be of user's interest. The search constraints are set starting from a user's profile specified by the user himself. The profile is then refined by observing how the user manages the proposed documents.
- *Internal Services.*
The *Interface Agent* translates requests concerning the activities and the archives of the user's organization for the *Internal Services Agent*.

Each agent has the same basic architecture, as described in the previous section, in which each component is specialized on the agent's jobs.

4 The Interface Agent

The *Interface Agent* is in charge of interacting with the user and of making transparent the other agents to him. Moreover it is able to understand the user's requests and translate them for the other agents. It is also in charge of coordinating the work of the other agents. It is the main reference point of the user.

The *Interface Agent* has different tasks that are crucial for the correct operation of the whole system:

- *Assisting the user in performing requests and compiling his profile.* The user does not need to be aware of: what is available on the network; how this information is structured and organized; where the repositories are localized; what retrieval services are at disposal. Once again it is the interface in charge of navigating the archives and presenting him what he is looking for.
- *Deducing the user's information needs by both communicating with him and observing his "behavior".* The agent observes the user's behavior and the current state of the world to deduce what actions are to be performed and how to modify the current user's profile. The user can or cannot be aware of the learning training of the agent(s) depending on both the user features and the processing step.
- *Translating the requests of the user and selecting the agent(s) able to solve his problem(s).* That allows the user to completely ignore the structure of the system he is

interacting with. Moreover he can also ignore how the system works. The user interacts with a *personalized* interface that knows how to satisfy his requests without bothering him.

- *Presenting and storing the retrieved data.* That avoids the user to know the different formats and how to manage a document to have a printable or showable version. The *Info Agent* deals each retrieved document according to its format and transforms it into a form the user can utilize.

The first interaction between the *Interface Agent* and the user occurs through a WWW window on which it is possible to select the basic tasks the user has at his disposal. At the present, the user can choose among three options:

- *Fill in a general profile.*
The user can specify his profile the first time he interacts with the *Interface Agent*. This profile will be automatically updated according to the user's behavior.
- *Set a specific query.*
The user can specify constraints for an *ad hoc* search. The *Info Agent* interprets this request for the *External Retrieval Agent*. Also these queries are used by the *Interface Agent* to update the user's profile.
- *FUB's Internal Service.*
The *Interface Agent* connects the user with internal archives and support services that are managed by the *Internal Services Agent*. The window called by the selection of this button is still handled by the *Interface Agent* that translates the user's request that becomes the content of a KQML message sent to the *Internal Services Agent*.

Each user's selection activates other interaction windows still managed by the *Interface Agent* even if they activate tasks performed by the other two agents. Our choice is that the only agent able to interact with the user is his personal assistant—in this case the *Interface Agent*—while the other agents receive from it the translation of the user's request and other data necessary to the accomplishment of the task. That makes the interaction easier for the user whose use of the computer is personalized: he always dialogues with the same entity and is not aware of who is performing the current task.

It is clear that the *Interface Agent* has to accomplish numerous and complex tasks and that it needs to have a picture of the current state of affairs. In particular, its knowledge component has a representation for:

- The user's world: the environment, the kind of relations the user has, his interests, his past history with respect to the retrieved documents, etc.
- The external world: its knowledge is either direct or indirect since it knows how to obtain what it needs or who can help it.

- The available agents and their capabilities: it coordinates a number of agents and knows what tasks they can accomplish, what resources they need, their availability.

The reasoning component processes the different knowledge sources in a way transparent to the user: the solution is directly dispatched to the user who is not implied in the management of the different functions involved by the agents. The *Info Agent* presents the results to the user according to his preferences. In the Section 4.1 the problem of the presentation of the results will be briefly discussed.

4.1 Dealing with User's Profiles

The *Interface Agent* keeps trace of the evolution of the user's interests maintaining a dynamic profile that takes the user's behavior into account. The specificity of the profile increases with the user's awareness about the available information and how to get it. The possibility of a relevance feedback is particularly important in the context of the final system: also other information agents, e.g., the *Mail Agent* [3] and the *News Agent* [1], not integrated in the current version, apply the relevance feedback [9] in dealing with profiles.

Using the user's profile, the *Interface Agent* charges specialized agents to navigate through the network hunting for information that could be of some interest for the user. In this way, the user can be alerted when new data that can concern his interest area(s) appear. The user can supply a first profile filling in a window that the agent presents to him at his first connection: Figure 2 shows a portion of such a window. The user can set different profiles each reflecting an interest area. Among the different preferences, the user can select the types of archives he is interested in, e.g., ftp, gopher, wais, etc. He can also set a *personal list* containing the sites in which documents of user's interest are found more frequently. The starting value for *personal list* is empty, afterwards the agent can modify it adding the more visited sites. Another choice regards how to treat the documents with respect to their format: each format is managed according to a default value by the agent even if the user can decide for a different treatment. For example a *dvi* document is printed by using the command *dvips* and shown by using *xdvi*, while a L^AT_EX document is shown or printed in a dvi format but stored in a postscript form.

A list of keywords identifies a possible set of documents for which the user can choose a particular action. Then he can specify the number of items he wants and if there is a time in which he prefers to activate the search.

The retrieved documents are shown to the user according to the preference values in the current profile.

As above mentioned, the *Info Agent* keeps trace of the user's behavior concerning the documents retrieved in both surfing and query modes. After each search cycle in the surfing mode, the retrieved documents are proposed to the user who can decide to refuse or accept each of them.

The rejected documents are stored in a database, named *Wastebasket*, and successively

compared with the sets of incoming documents in order to refine the boundaries of the search.

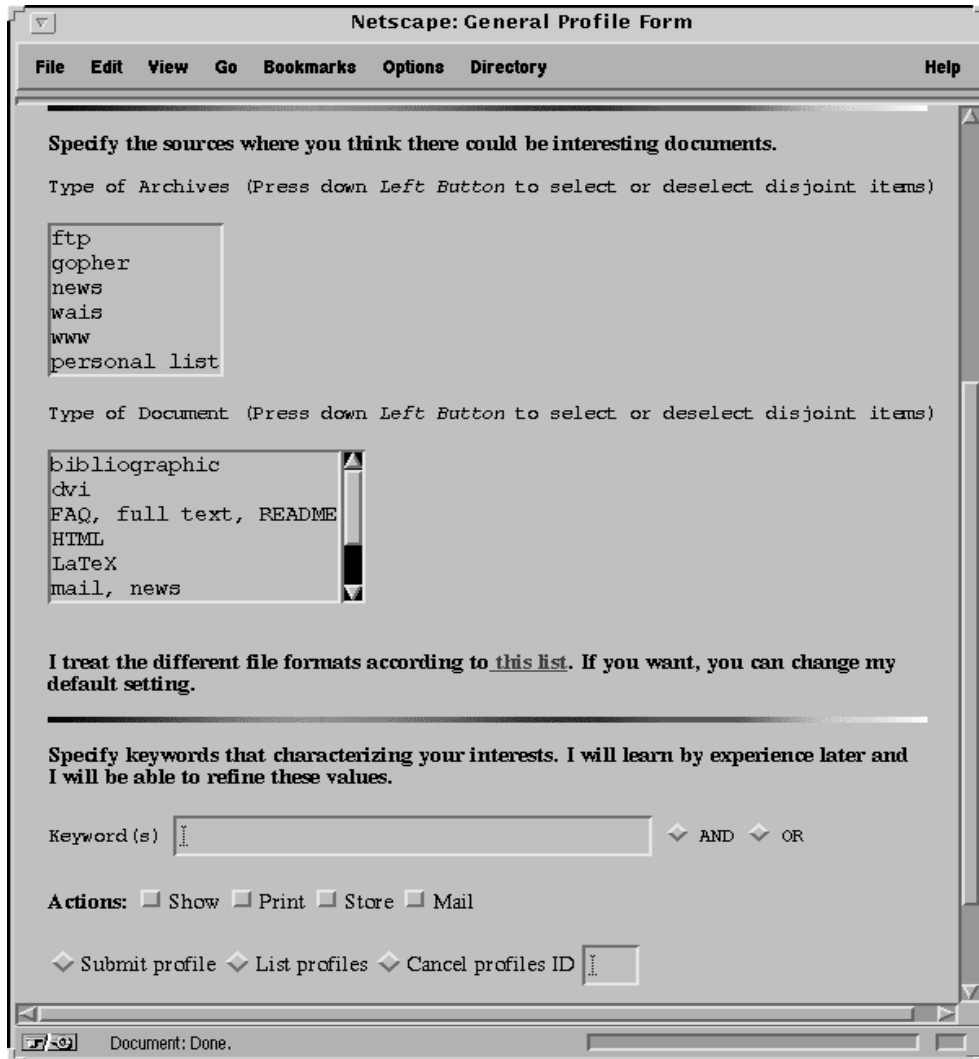


Figure 2: A portion of the window for setting a user's profile

The two sets of documents are matched using information retrieval techniques. If items in the incoming set are found similar to some of the rejected documents, the agent discards the former. As a consequence the documents proposed to the user are closer to his actual interests. We are studying a method to definitely store these changes in the profiles with the aim of cutting the search from the beginning. The user will be able to control the modifications.

In the query mode, the user's requests are also used to refine the profile. The rejected documents are added to the *Wastebasket*, while for each query a profile is extracted from the set of accepted items that the agent adds to the profiles DB.

5 The External Retrieval Agent

It is the agent devoted to navigate external archives, catalogs, Web repositories to search the information requested by the user's *Interface Agent*. It can work in two modalities, surfing and searching, according to the directions of the *Interface Agent* that sends it a KQML message whose body contains the features of the task to be performed.

This agent uses a public domain tool called HARVEST [8] to perform its search. HARVEST is designed to manage any type of resource across the Internet: it is able to structure a large range of information sources, e.g., postscript documents or software available through ftp. The system consists of two main components, the Gatherer and the Broker. The Gatherer collects data across the network by utilizing any possible access method (FTP, Gopher, Wais, News, HTTP, etc): then it structures them by extracting any relevant feature. For example, a technical report is characterized by its title, author(s), date, etc., while a software is characterized by the language, release, etc. Each retrieved item is structured by using the SOIF (Summary Object Interchange Format) format that transform it into a set of attribute-value pairs. The Gatherer builds an internal database containing all the collected items. The Broker uses the databases collected by several Gatherers to satisfy a request and offers a language to performs structured queries. The search is optimized since the Broker uses an indexed representation of the Gatherers' databases.

Due to the agent architecture it is easy to substitute HARVEST with another tool or to add another tool or method to it.

5.1 The surfing mode

In the surfing mode, the *External Retrieval Agent* interacts only with the HARVEST's Broker component. It translates the KQML message coming from the *Interface Agent* in a query into the Broker's language. The user is not aware of the utilization of the HARVEST system since the agent hides the operations it accomplishes. The Broker inspects the collection of databases to satisfy the query. It can also ask other Brokers holding different databases: in turn, these Brokers can pass the request to other Brokers until arriving to a Broker connected to a Gatherer working as a provider host. It is to remember that the search is always performed on the indexed sets.

The retrieved documents are passed to the *Interface Agent* that put them at user's disposal according to what described in Section 4.1.

5.2 The query mode

In the query mode the user asks the agent to find a particular item corresponding to a description. The user can specify the references (author, title, etc.) in a WWW window. Then he can introduce further constraints by specifying the number of items, the type of search and when to start it.

The user's requests are utilized to refine his profile. Each request, the set of retrieved documents—both accepted and rejected—and the actions performed on them are translated into user's profile and added to the profiles database.

The search is always performed with the support of the Broker: in this case, only the databases relevant for the query are inspected. The agent is able to inspect not only HARVEST's Brokers but also different type of repositories as Lycos, WAIS, etc. The *External Retrieval Agent* holds an exhaustive collection of the sites and their characterization, i.e., which type of documents they maintain, in which format, etc.

In a future version, the *External Retrieval Agent* will be also able to accomplish the Gatherer's functions in order to generate an internal database to take into account sites that are not included in its Broker's collection. That will allow the agent to optimize its searches that are performed on indexed database; moreover, that also diminishes the traffic on the network.

The retrieved documents are then passed to the *Info Agent* that manages them as said in the previous section.

6 The Internal Services Agent

The *Internal Services Agent* is devoted to deal with internal archives of different nature and to support the user in dealing with some bureaucratic matters. It is called by the *Interface Agent* that translates a user's request for it. This kind of agent is strictly depending on the information structure of the represented site. Nevertheless its architecture reflects the general principles inspiring the agent modeling. The knowledge about the organization's structure is part of the particular instance of the agent. Obviously our implementation concerns the FUB internal archives and organization.

The actions the agent can perform are listed in the following:

- *Checking for resource availability*

The user can ask both generic questions, e.g., *Is there a free audio board for a Sparc station 20?*, and specific questions, e.g., *Why the printer Lpr5.1 is not working?*. The availability of the software tools and some types of hardware instruments is given by searching in an internal database; for the currently used machines, i.e., printers, computers, scanners, etc., the resource manager updates the database when a new situation occurs. It is also possible to directly send an e-mail message containing the request of getting new resources to the person in charge of acquiring them.

- *Searching a FUB technical report*
 Since the FUB—as other research organizations—produces hundred reports every years, it is not always easy to find what one is looking for. The user can ask the agent to search the reports that correspond to given references in the internal database. Then the retrieved reports are shown to the user who can decide whether listing, printing or storing them.
- *Searching in the FUB Library*
 Since the FUB library is “distributed”, i.e., each research scientist stores a number of books, journals, etc., finding what are you looking for is not a easy task. One has to find the person who has the document, then call him/her and eventually verify the availability. This task is accomplished by the agent that retrieves the person, sends him/her an email message and then communicates to the user when and where getting the document.
- *Searching data on Annual Reports and Future Plans*
 The agent helps the user to navigate the collection of annual reviews and descriptions of incoming projects of the FUB.
- *Searching in the Personnel Database*
 If the user wants to contact a FUB person, this is the way. The FUB people are stored in a database containing all the relevant and public data concerning them. The agent can also contact them directly via e-mail.
- *Helping in filling in forms*
 The agent supports the user in compiling internal forms concerning both administrative and research matters, for example requests for funding. The form is then automatically sent to the person in charge.

The data are spread into heterogeneous and distributed databases, the formats of which are also different. The *Internal Services Agent* translates the queries according to the type of the database, then it returns the retrieved data to the *Interface Agent*.

7 Conclusive Remarks

In this paper an architecture has been presented to support a user in searching through both local and remote distributed databases, and to assist him in dealing with administrative issues. Characterizing issues of our approach are:

1. the autonomy of the agents concerning how to accomplish the user’s goals;
2. the observation of the user’s behavior and the role of an automatic user feedback.

Concerning the first issue, the system is provided with an intelligent and active interface that completely hides to the users the utilization of different software tools used to solve the users' problems. The mechanization of some functionalities allows the user to deal with complex tools without having to learn them. The *Info Agent* helps the user to face with the problem of the information overloading: in fact the agent substitutes the user in boring and time-consuming operations. Moreover, the *Info Agent* has an active role in searching and proposing documents without any precise request by the user.

As far as the second issue is concerned, the *Info Agent* is also able to modify its actions according to the past history of the user. The agent starts with a rough user's profile that is automatically refined without the user had to worry about. A relevance feedback functionality is utilized by two information agents not integrated yet in the current version of the system, the *Mail Agent* and the *News Agent*. The type of feedback used depends on the expertise and the features of the users. At present we are experimenting three types of feedback: the final choice will depend on the result of such a test. Probably, the best solution could be to provide different types of feedback so that each user can choose the most suitable to her preferences.

Acknowledgments

This work was carried out in the framework of the agreement between Telecom Italia and the Fondazione Ugo Bordoni. Daniela D'Aloisi also collaborates with the IP-CNR of the National Research Council of Italy in the framework of an agreement between the FUB and the IP-CNR.

References

- [1] Amati, G., D'Aloisi, D., Giannini, V., A framework for dealing with email and news messages. Proceeding of AICA Conference, Cagliari (Italy), September 27-29, 1995, 353-360.
- [2] Brown, C., Gasser, L., O'Leary, D., Sangster, A., AI on the WWW: Supply and Demand Agents. *IEEE Expert*, Vol.10, No.4, August 1995, 50-55.
- [3] Cesta, A., D'Aloisi, D., Giannini, V., Active Interfaces for Software Tools. In Y.Anzai, K.Ogawa and H.Mori (editors), *Symbiosis of Human and Artifact* (Proceedings of the 6th International Conference on Human-Computer Interaction, Tokyo, July 9-14, 1995), Elsevier Science B.V., pp.225-230, 1995.
- [4] Cesta, A., D'Aloisi, D., A General Architecture for Software Agents. In preparation.

- [5] Etzioni, O., et al. (eds.), Working Notes of the AAAI Spring Symposium on Software Agent. AAAI Press, 1995.
- [6] Etzioni, O., Weld, D.S., A Softbot-Based Interface to the Internet. *Communication of the ACM*, Vol.37, No., July 1994, 72-76.
- [7] Finin, T., Weber, J., et al., Specification of the KQLM Agent-Communication Language. DRAFT, February 1994.
- [8] Hardy, D.R., Schwartz, M.F., Wessels, D., Harvest: Effective Use of Internet Information (Harvest User's Manual, Version 1.2). Technical Report CU-CS-743-94, University of Colorado at Boulder, April 1995.
- [9] Harman, D., Relevance Feedback Revisited. In N.Belkin, P.Ingwensen, M.Pejtensen, (eds.), Proceedings of the 15th Annual International Conference on Research and Development in Information Retrieval. ACM Press, 1992, 1-10.
- [10] Lashkari, Y., Metral, M., Maes, P., Collaborative Interface Agents. Proceedings of the 12th National Conference on Artificial Intelligence (AAAI 94), Seattle (WA), 1994, 444-449.
- [11] Steiner, D.D., Mahling, D.E., Haugeneder, H., Human Computer Cooperative Work. In H.Huhns (ed.), Proceedings of the 10th International Workshop on Distributed Artificial Intelligence.
- [12] Thomas, C.G., BASAR: A framework for integrating agents in the World Wide Web. *IEEE Computer*, Vol.28, No.5, May 1995, 84-86.