Supporting Information Consumers by Search Agents in the World-Wide Web

Christoph G. Thomas, and Reinhard Oppermann

HCI Research Division (FIT.MMK)
GMD - German National Research Center for Information Technology
D-53754 Sankt Augustin, Germany
Tel.: +49 2241 14 2640/2703, Fax: +49 2241 14 2065
{Christoph.Thomas, Reinhard.Oppermann}@gmd.de

Abstract. Due to emerging technologies a wide area of network services has grown up around the Internet: tools like World-Wide Web provide a massive amount of information. Being lost in space and overloaded with information are two problems information consumers confront: there is more information out there than a single consumer can manage. In consequence, finding information can be frustrating and time-consuming: users need active support to determine *if* potentially useful information exists, *where* the information is located, *how* to retrieve the information when it is located, and *how* to use the information when it is retrieved.

To address and overcome problems of the WWW, we have designed and implemented a framework to integrate agents into the use of the WWW. The agents filter information, initiate communication, monitor events, and perform tasks. The agents rely on usage profiles to adapt their assistance to specific users.

Keywords: User modeling, Intelligent user interfaces, Guided interaction and intelligent agents, architectures and frameworks, Adaptable and adaptive interaction

1. INTRODUCTION

Due to emerging technologies a wide area of network services has grown up around the Internet: tools like World-Wide Web (WWW) provide a massive amount of information [Krol 94]. Being lost in space and overloaded with information [Schick 90] are two problems information consumers confront: there is more information out there than a single consumer can manage. In consequence, new strategies are needed to deal with information spaces such as the WWW: consumers need active support to determine *if* potentially useful information exists, *where* the information is located, *how* to retrieve the information when it is located, and *how* to use the information when it is retrieved.

One solution is that software agents actively support a user. The kind of tasks that agents might perform in such systems is ranking from supporting the navigation and browsing process, making the information retrieval easier, doing sorting and organizing (indexing) jobs for the user, and filtering information in large data bases [Laurel 90].

We explore the usability and usefulness of that solution to reduce the information overload problem. The usability will be improved when the existing functionality can be adapted to the user's specific needs. And the usefulness will be improved by extending WWW's functionality.

Based on empirical studies of users and their problems in dealing with large information spaces we have developed the system BASAR (Building Agents Supporting Adaptive Retrieval).

2. FEATURES OF BASAR

BASAR embeds software agents acting as personal assistants in the WWW. These agents offer support by (a) adapting searching and filtering, and by (b) reducing and restructuring the access space to active views as task-dependent personal information spaces.

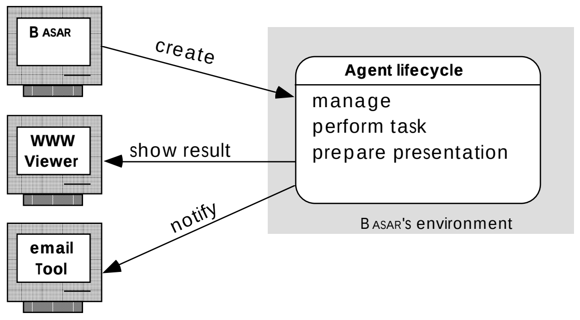


Figure 1: Interfaces of BASAR

BASAR provides users with an interface to create and manage agents—this is done with windows and dialog boxes in the BASAR environment. Agents present their task results in a WWW viewer, for example MOSAIC, and agents communicate through the user's preferred method—for example, via email if the user is absent.

BASAR embodies the following characteristics [Thomas 95].

_ **Software Agents**: BASAR provides users with an environment for creating and using agents that actively support them in locating, relocating, and filtering information they desire (Figure 1). BASAR comes with a set of pre-defined agents (see Table 1); it is up to the user to choose one of these agents from a menu or to create new agents by using an agent editor. Active agents are listed in a window that informs the user about their ongoing tasks.

_ **Usage Profile**: BASAR builds a model of the user (preferences, interests, and tasks) using both explicit (asking the user) and implicit (observing the user) modeling techniques. This is done to adapt the assistance specific to each user [Fischer 85, Krogsæter 94].

_ **Active Views**: BASAR provides users with a concept for creating personal information spaces along a semantical meaning that is independent of the WWW-viewer (such as MOSAIC or NETSCAPE). An active view is defined as a set of bookmarks belonging to the same header in combination with a set of agents attached to it: <active view> = <bookmark header> + <bookmark links> + <agents>.
The agents are responsible for keeping an active view manageable by adding, updating, and removing information links. For example, agents make suggestions to add information links to a view if these links have been visited often by the user in the past, agents notify the user when information has been updated, or agents suggest removing links that have become invalid. An active view is user-specific; its description as part of the usage profile allows analysis of user's actions on that view, for example, searching for, deleting, or adding links.

BASAR has been implemented for the WWW client MOSAIC under X-Windows in VisualWorks 2.0. The user normally interacts with WWW and MOSAIC. BASAR has access to the functionality of MOSAIC, to the logfile of user actions, the usage profile, and to the distributed Web space.

The user of BASAR deals with active views; agents work on these views, and the usage profile makes the assistance of the agents specific to each user. To briefly summarize, these characteristics will improve the usability and the usefulness of the WWW as follows:

_ Users get active support from the system; for example, agents notify the user about old, new or updated information relevant to an active view. So far, hotlists and bookmarks are passive repositories for information (links).

_ Users can delegate tasks to agents; for example, users can ask agents to look for new links in the result list of a search process that are not stored in the user's personal information space. So far, users have to do that by hand.

_ Users can ask agents to perform periodical tasks; for example, agents can look for new information on a specific server every week. So far, users have to do that themselves.

_ Users get uniform access to search engines with a knowledge base in the background describing how the different search engines work, how to call them, the best time to call them, and how relevant their results were in the past. So far, users have to remember that every time they call a search engine.

Delegating tasks and supporting the browsing and searching process enable a user to (a) use less time to search the same information space, or (b) search a larger space in the same time. These represent two important strategies to reduce the information overload problem.

3. A SCENARIO: USING SEARCH ENGINES

The following scenario illustrates our ideas and may help to clarify the underlying conceptual work. Suppose, a user's task is to create a personal information space called ”lifelong learning.” We assume that this information space will grow dynamically over time and become a permanent information repositorium of all useful WWW links relevant to the area of lifelong learning, including links to research institutions, grants, people, papers, and initiatives.

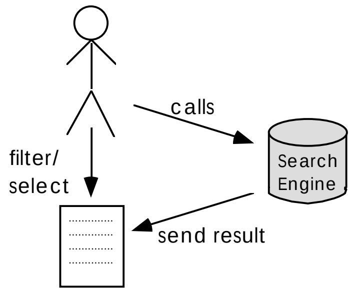


Figure 2: Searching in the World-Wide Web without BASAR

3.1 How Information Consumers Use Search Engines

One usual way to start with the WWW is to call one (or more) search engines that are requested to return links relevant to a given search key. It is then the user's task to filter the result (Table 1) by hand (Figure 2). As search engines are still the most popular tools to find information in the Web space, we were interested in how people use these tools. Therefore, we developed a questionnaire about search engines and distributed it to a GMD newsgroup and to *de.comp.infosystems*. The most important results (based on 22 replies) are:

Users tend to use the same search engines again and again no matter what they are looking for, e.g. persons, institutions, software, publications, etc.—despite the well-known fact that different search engines use different techniques and algorithms to analyze WWW pages. Once users get used to one or more specific search engines they stay with them.

Three-fourths of the respondents repeat a search some time later on with the same keywords. This is done for almost two reasons: either people want to know whether some new information relevant to the search key has been added to the World-Wide Web or people try to find some lost information link again.

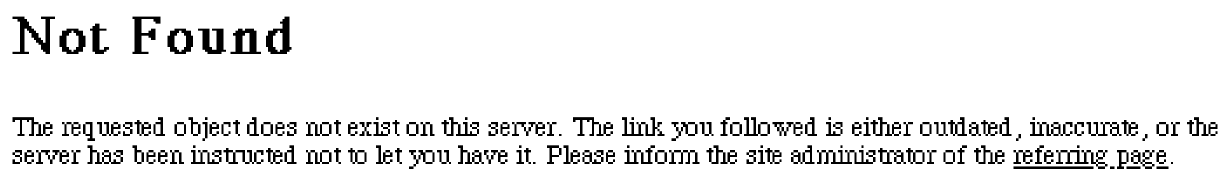
Once a search engine returns a results, it is up to the user to select relevant information links and interpret the results what depends on user's time and experience. Most used selection criteria for links are, for example, the order suggested by the search engine, a geographical order, or the reference string. Users tend to select links in search results in different manner than they are displayed to the user.

Most users store results from search engines that seem to be relevant for their purposes as bookmarks in their bookmark list. Managing bookmarks and bookmark lists is another very important task for information consumers to deal with information overload, for a discussion see [Thomas 96].

But difficulties that come with search engines are not only restricted on how to use the results, they even start when a search engine has been selected.

_ **Accessibility**: Sometimes search engines are not accessible, either due to a shutdown, too much net traffic in general, too much network accesses to the search engine at query time, or other facts. In consequence, the user has to call the search engines again and again until success is achieved.

_ **Validity**: When results are returned, users need to know what links are still valid (links becoming invalid over time is a general problem for the WWW). For example, a (randomly) selected entry from the results of the search engine RBSE's URL database leads to an unexpected error:


Not Found

The requested object does not exist on this server. The link you followed is either outdated, inaccurate, or the server has been instructed not to let you have it. Please inform the site administrator of the referring page.

_ **Multiplicity**: Users need to know which links appear in more than one search result if more than one search engine was involved in the search process. In consequence, users have to compare the different results by hand and remove duplicates.

_ **Already known**: Users wants to know which links are already stored in their personal information space (e.g. described through bookmarks). In consequence, users have to compare the links in their personal information space with the new ones.

_ **Minor relevance**: Users would like to easily identify the links they have already visited and considered earlier to be of no or minor interest. In consequence, users spend annoying time looking at sites that are not worth looking at.

_ **Iterative process**: There is no support in doing a search periodically. In consequence, users have to call search engines explicitly every time they want to update their personal information space.

Name of Search Engine	**Result for search key "Lifelong Learning"**
NIKOS	2
RBSE's URL database	100 (number restricted by user)
Jumpstation II	0
Lycos	Found 14434 documents matching at least one search term. Printing only the first 15 of 14434 documents ...
WebCrawler	found 102 documents, returned 25

Table 1: Results of Search

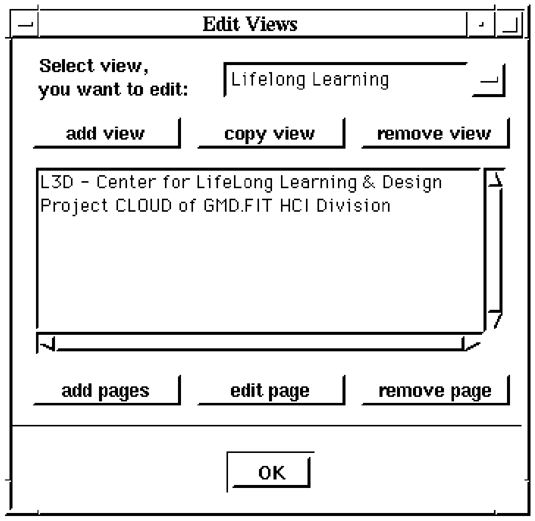
This table shows the result of the use of five different search engines that have been called with the keyword "lifelong learning." The quality and the quantity of the results is very different among the search engines. This small experiment was done at the end of May 1995.

The user has to be aware of these problems when using search engines. To sum up, these tests confirmed that users have difficulties in making efficient use of search engines.

3.2 WWW—with BASAR

In consequence, we support the search process by introducing agents, usage profiles, and active views. First, the user defines an active view called "lifelong learning." This is done by creating the new active view in a simple dialog box, see Figure 3.

The call and the results of a search process are stored in the usage profile together with the actions the user performed on these results. The search agent evaluates the success of a specific search and uses it later for another search task (the support of a search agent improves with the number of its usages). The extracted information is used when the user calls the search agent again. If the search engine returns a result, the search agent filters all the information links according to the usage profile, for example, telling that the user is most interested in information links coming from *.edu* or *.com* server sites.



Lifelong Learning

L3D - Center for LifeLong Learning & Design
Project CLOUD of GMD.FIT HCI Division

Figure 4: Creating the Active View ”Lifelong Learning”

The creation is done in BASAR ‘s ”Edit View” window by selecting the ”add view” button that asks for the new view’s name. Once created, links are added to that view by selecting ”add pages.” In this example, the new view starts with two links.

Second, the user creates an agent to work on that view. The type of agent that is relevant for this scenario is called a *search agent* (which is one of the predefined agents that come with BASAR, see Table 2). The search agent mediates between different search engines and the user (Figure 4). Its behavior is influenced by the information contained either in the logfile or in the usage profile.

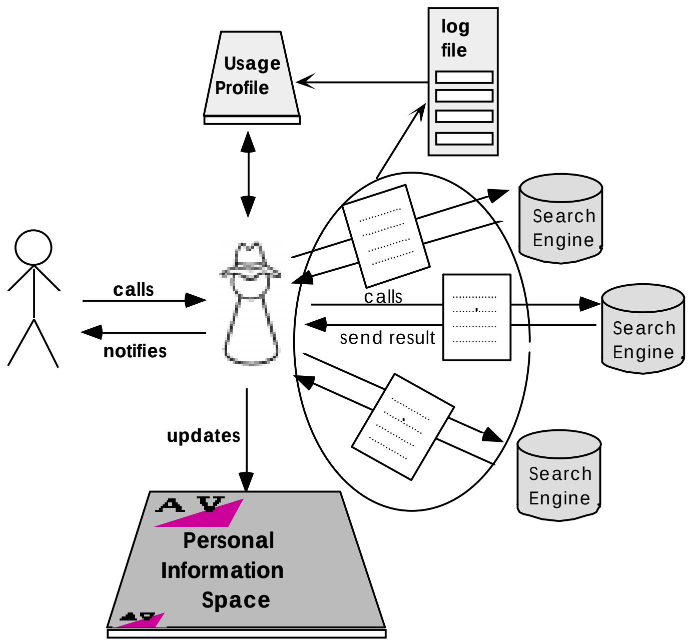
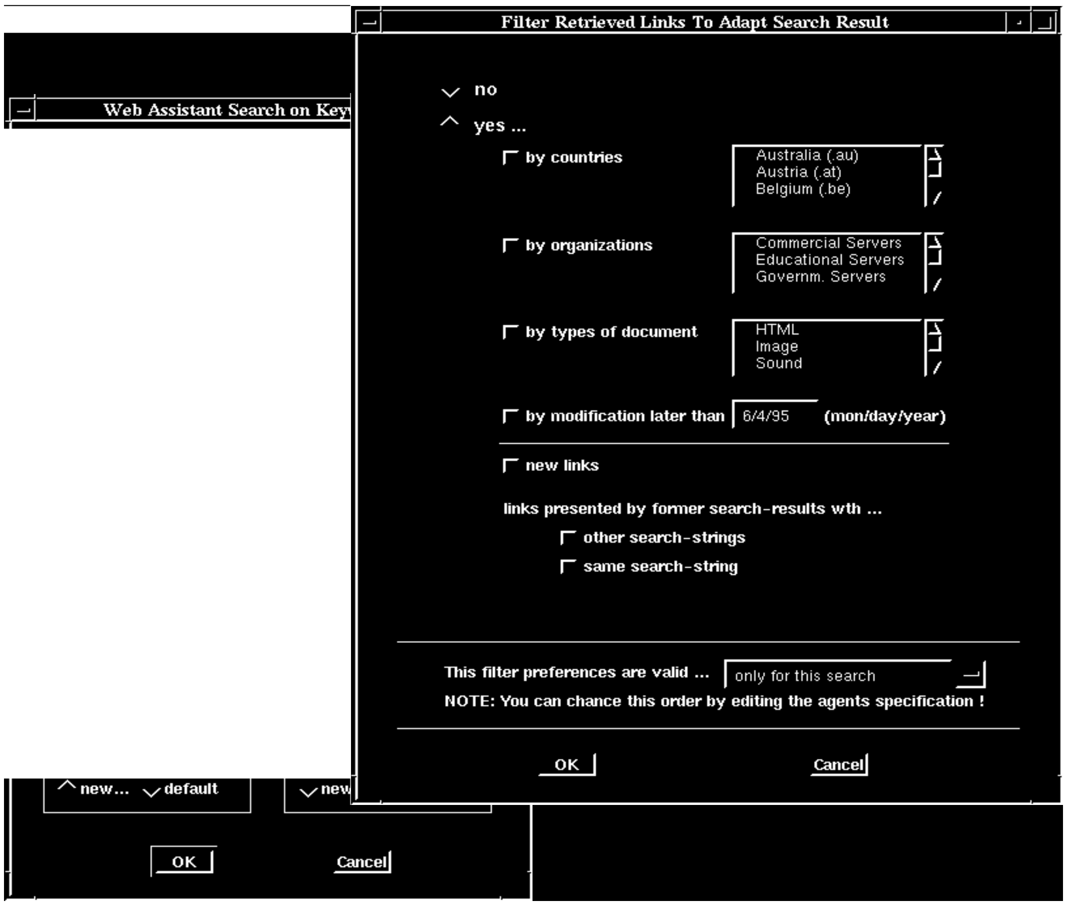


Figure 4: Searching in the World-Wide Web with BASAR

The logfile contains the global history, that is the set of links the user has visited. But the logfile does not know anything about the semantical meaning of a WWW page, such as that it describes the result of a search process and may contain a set of relevant links asked for by the user. And even the logfile does not contain the information that a search process has been started. This is the point where the usage profile with its three parts (domain-specifics, task-specifics, and user-specifics) comes in, see Figure 6.

The domain-specifics part (which is the network knowledge base) tells the agent (a) how to call search engines, (b) when is the best time to call them, and (c) to try to call a search engine later on again until success if the request fails. This reduces the ”accessibility” problem mentioned above.



Filter Retrieved Links To Adapt Search Result

Web Assistant Search on Key

no

yes ...

by countries: Australia (.au), Austria (.at), Belgium (.be)

by organizations: Commercial Servers, Educational Servers, Governm. Servers

by types of document: HTML, Image, Sound

by modification later than 6/4/95 (mon/day/year)

new links

links presented by former search-results wth ...

other search-strings

same search-string

This filter preferences are valid ... only for this search

NOTE: You can chance this order by editing the agents specification !

Figure 5. Creating a Search Agent

The dialog box on the left provides users with a common interface to the most popular search engines. The agent will call the selected search engines until reaching success. The window on the right pops up after selecting the button ”filter preferences using new.” It presents a table of the user’s preferences on how to sort the search results.

The task-specifics part tells the agent (a) to delete multiple occurrences of the same links if the user has selected more than one search engine (reduces the problem of ”multiplicity”), and (b) to compare the result links with the links already stored in the user’s personal information space (reduces the ”already known” problem).

Due to an entry in the user-specific part of the usage profile, the agent is able to evaluate the relevance of former search result links for the user: if the user did look at a link but did not store it, it is supposed to be of minor relevance.

And last but not least, search agents, as all the other types of agents in our system, can perform their tasks periodically, which reduces the ”iterative process” problem.

The results of a search process can be ordered by different criteria, such as country code, server sites, document type, or visited pages. The type of ordering is specified either implicitly by analyzing how the user dealed with search results of former searcher or, explicitly, by editing a dialog window from the usage profile, see Figure 5.

3. THE CONCEPTUAL FRAMEWORK

We have adopted *indirect management* [Kay 90], see Figure 6, as the fundamental model of interaction because it integrates agents supporting users in doing their tasks at hand. Both users and agents initiate communication, monitor events, and perform tasks instead of having unidirectional interaction via commands and/or direct manipulation.

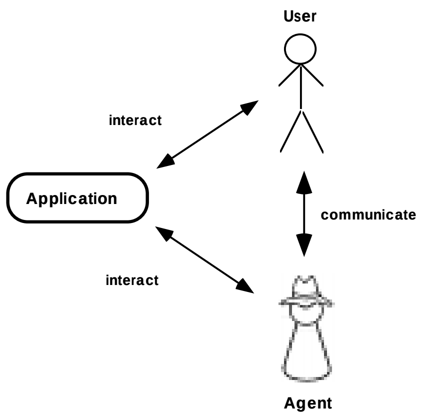


Figure 6: Indirect Management

The agent acts as an additional component to the user and the application. It depends on agent's tasks whether it is modeled more closely to the user, to the application or to the interface.

The purpose of the agent is to assist the user and to adapt this assistance to be specific to each user. A user should be able to delegate (sub-)tasks to the agent, whereas the agent should be able to infer the need for user-specific support during a problem-solving or task-performing process.

This model leads to a conceptual framework for integrating software agents in the WWW, see Figure 7. An ”ideal” agent support (i.e. improving both usability and usefulness) needs knowledge about the interaction between the user and the system (for user-specific support), the functionality of the system itself (for task-specific support) and the problem solving process within the application domain of system (for domain-specific support). These different types of knowledge are covered in the usage profile.

For the purpose of BASAR, we have created three specific classes of an agent: *interface* agents (that know to communicate with the user), *task* agents (that know to perform a task), and *network* agents (that know to communicate across the network).

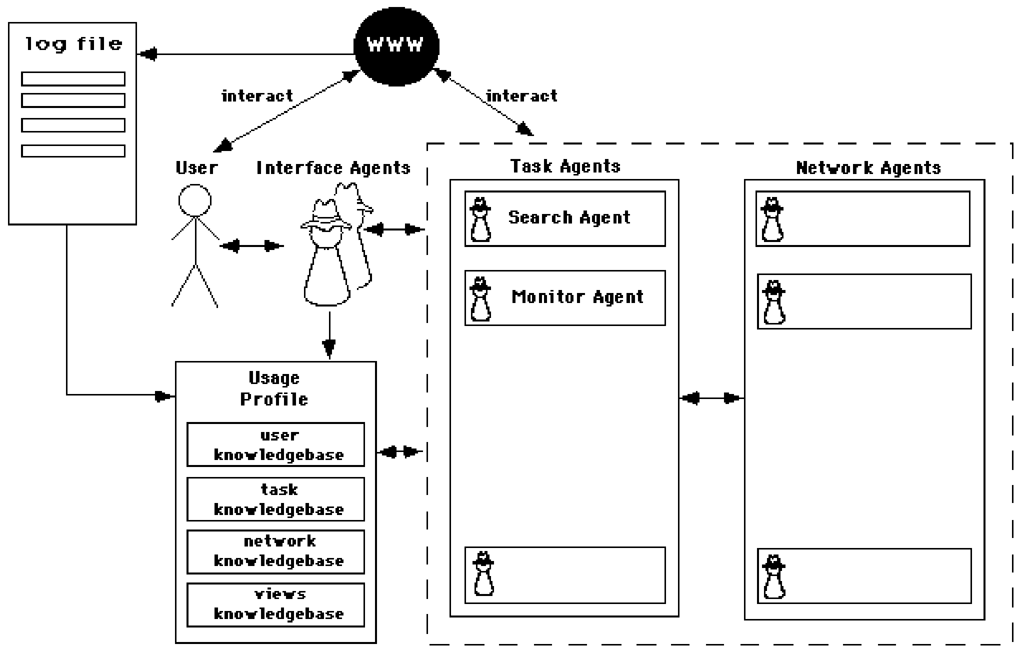


Figure 7: Basar Architecture

Agents have access to both the functionality and the resources of the application by its programmer interface. Also, agents have to communicate with the user. This is done through the agent's user interface, which extends the original user interface of the application.

Interface agents mediate between the user and task or network agents, communicating through the user's preferred method—for example, via email if the user is absent or via a blinking icon if the user is present. If a network or a task agent wants to contact the user, it requests an interface agent. The interface agent's behavior—that is, allowing, deferring, or denying contact—is defined and determined by the usage profile. A subclass of an interface agent is the *view agent*, which is responsible for presenting an active view to the user. The view agent constructs a default representation of the active view by getting a view description from the usage profile.

Task agents support adaptive filtering, the creation of active views, and locating and accessing relevant information. Some examples of build-in task agents are listed in Table 2.

Network agents are implemented on a client/server architecture. Based on their knowledge of the network—location of search engines, available server, time zones, different types of server sites such as .com, .edu, .de—network agents ship task agents to appropriate WWW sites.

The user interface of BASAR hides the distinction between interface, task, and network agents from the user; the user simply interacts with agents through email, icons, dialog boxes, or within active views. The top-level window of BASAR informs the user about all active agents with a short description of their task. It also provides the user with functionality to create active views (see, for example, Figure 3), to edit the usage profile, and to create agents through an agent editor, for example, giving the agent a name, specifying whether the agent should do a single or a periodical task, selecting the task to perform, and allocating an active view to the new agent for presenting the results.

5. RELATED WORK

Software agents technology is on its way to being used in commercial products, for example, for workflow and network management, in messaging, and in information retrieval [Guilfoyle 94]. For example, Apple's APPLESEARCH software enables the creation of personal search agents ("Reporters") to search incoming mail messages and documents from on-line services [Roesler 94], or TELESCRIPT lets users send executable programs in the form of agents through the network [Wayner 95].

Predefined Agents	**Purpose**
clean-up agent	takes the hotlist, looks for dead links, asks the user to delete hotlist entries not selected for a period of <n> months
search agent	supports users in the use of search engines and their results
filter agent	compacts information and adapts it to user's need according to the usage profile
monitor bbs agent (bbs = bulletin blackboard system)	this agent monitors a Web page used as a blackboard for an active group view

Table 2: Built-in Task Agents

In the WWW, search engines, also called Internet Agents [Indermaur 95], are the first attempts to integrate agent technology. But, as seen in this paper, their concept differs in many ways from the concept of the assistant agents of interest to us. Another area with agent technology in the WWW has been named "collaborative information filtering," a technique to support information consumers in finding relevant information by making use of what others have already found and evaluated [Maltz 95]. For example, HOMR [Shardanand 95] is a collaborative information-filtering system based on learning agent technology [Maes 94].

In contrast to search engines, systems such as HOMR build a user profile, called an interest profile, and make personalized recommendations based upon values assigned by other people with similar tastes. Such systems can be used for any database, which may be of great advantage on the one side, but on the other side they do not contain, like BASAR, WWW-specific knowledge, analyze user's dialog history, and build a usage profile that supports the managing of WWW personal information spaces. To our knowledge, BASAR is a unique attempt to integrate agent technology with user modeling techniques into the WWW.

6. DISCUSSION

BASAR is the newest prototype in our ongoing research efforts to explore the embedding of intelligent agents and user modeling techniques into domain-oriented systems. BASAR continues the work of an earlier implemented system, FLEXCEL [Thomas 93, Oppermann 94], as an adaptive user interface extension for the spreadsheet program Excel® from Microsoft.

In our conceptual framework, we consequently have adopted indirect management as the fundamental metaphor for human-computer communication, which raises numerous conceptual, technical, and social issues. These issues are a consequence of the mixed-initiative dialogs made possible by the agents. With BASAR, we are investigating these issues for the WWW as a testing substrate of a new type of information space. The conceptual issues we are investigating with BASAR include control of initiative and intervention, and focus of attention. The technical problems include the embedding of agents in the WWW as an existing information space, their communication with WWW clients such as MOSAIC, the use of existing WWW tools such as search engines, user manipulation of agents through an agent editor, activation of agents, and presentation of agents and their results. Social issues addressed by our research include the new role distribution between user and agents, namely, the embedding of agents in new types of information systems that complement information access with information delivery.

7. FUTURE WORK

The future work on BASAR has mainly two directions: (a) identifying its shortcomings by assessments and empirical evaluations, and (b) extending the concepts of active views and agents to support groups of users.

One shortcoming of our prototype is that it needs two systems, a WWW viewer such as MOSAIC and a SMALLTALK environment for the creation and control of the agents and the active views. A much nicer idea is to make the basic features of BASAR directly accessible through any viewer of the WWW, as is done, for example, in HOMR. This could simplify the installation and loading of BASAR and its communication with agents and views.

Until now, BASAR has been for single users. To make it suited for a group of users we started to extend active views to *active group views*. For example, the view on ”lifelong learning” could be used by the members of different research groups as a joint information repositorium.

REFERENCES

[Fischer 85] G. Fischer, A.C. Lemke, Th. Schwab, Knowledge-Based Help Systems. Human Factors in Computing Systems, CHI’85 Conference Proceedings (San Francisco, CA), ACM, New York, 1985, 161-167.

[Guilfoyle 94] C. Guilfoyle, E. Warner, Intelligent Agents: the New Revolution in Software. Ovum Limited, London, UK, 1994.

[Indermaur 1995] K. Indermaur, Baby Steps. BYTE, March 1995, 97-104.

[Kay 1990] A. Kay, User Interface: A Personal View. In: B. Laurel (ed): The Art of Human-Computer Interface Design, Addison Wesley Publishing Company, Inc., 1990.

[Krogsæter 94] M. Krogsæter, R Oppermann, C.G. Thomas, A User Interface Integrating Adaptability and Adaptivity. In: R. Oppermann (ed.): Adaptive User Support. Lawrence Erlbaum Associates 1994, 97-125.

[Krol 94] E. Krol, The Whole Internet. User's Guide & Catalog. Second Edition. O'Reilly & Associates, Inc., Sebastopol, CA, 1994.

[Laurel 90] B. Laurel, Interface Agents: Metaphors with Character. In: B. Laurel (ed): The Art of Human-Computer Interface Design, Addison Wesley Publishing Company, Inc., 1990.

[Maes 94] P. Maes, Agents that Reduce Work and Information Overload. CACM, July 1994, 37, 7, 3-40.

[Maltz 95] D. Maltz, K. Ehrlich, Pointing The Way: Active Collaborative Filtering. Mosaic of Creativity, CHI'95 Conference Proceedings, ACM, New York, 1995, 202-209.

[Oppermann 94] R. Oppermann (ed.), Adaptive User Support. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1994.

[Roesler 94] M. Roesler, D.T. Hawkins, Intelligent Agents - Software Servants For An Electronic Information World (and More!). ONLINE, 18, 4, 1994, 1-32.

[Schick 90] A.G. Schick, L.A. Gordon, S. Haka, Information Overload: A Temporal Approach. Accounting Organizations and Society, 15, 3, 1990, 199-220.

[Shardanand 95] U. Shardanand, P. Maes, Social Information Filtering: Algorithms for Automating ”Word of Mouth”. Mosaic of Creativity, CHI'95 Conference Proceedings, ACM, New York, 1995, 210-217.

[Thomas 93] C.G. Thomas, Design, Implementation, and Evaluation of an Adaptive User Interface. Knowledge-Based Systems. Special Issue on Intelligent Interfaces. 6, 4, 1993, 230-238.

[Thomas 95] C.G. Thomas, Basar: A framework for integrating agents in the WorldWide Web. IEEE Computer, 28, 5, 1995, 84-86.

[Thomas 96] C.G. Thomas, G. Fischer, Active Views: Managing Personal Information Spaces in the World-Wide Web. submitted to CHI´96, Vancouver, Canada, April 1996.

[Wayner 95] P. Wayner, Free Agents. BYTE, March 1995, 105-114.