Using Agents to Improve the Usability and Usefulness of the World-Wide Web

*Christoph G. Thomas*1,2 *& Gerhard Fischer3*

Abstract: The World-Wide Web (WWW) has emerged as a new type of information space. Its lack of central control mechanisms leads to many interesting new features but at the same time has the potential danger that users can drown in irrelevant information.

To address and overcome problems of global information spaces such as 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.

In this paper we describe the unique challenges of a WWW-type information space, present the design rationale and conceptual framework behind our work, and illustrate with scenarios the use of the system.

Keywords: intelligent/adaptive user interfaces, information retrieval and information filtering, information lifecycle, personal information spaces, information delivery, agents, usage profiles, World-Wide Web

1HCI Research Division (FIT.MMK)
GMD - German National Research Center for Information Technology
D-53754 Sankt Augustin, Germany
Email: Christoph.Thomas@gmd.de

L3D - Center for LifeLong Learning and Design
University of Colorado at Boulder
Boulder, CO 80309-0430, USA
Phone: (303) 492-1502, Fax: (303) 492-2844
Email: gerhard@cs.colorado.edu

1. Introduction

The WWW has emerged as a new type of information space. Its lack of central control mechanisms leads to many new interesting features but at the same time has the potential danger that users can drown in irrelevant information. Being lost in space and overloaded with information [Schick, Gordon & Haka 1990] are two problems users confront: there is more information out there than a single user can manage.

The potential benefits of the WWW will not be realized if users cannot retrieve information easily and efficiently. The principal techniques that have long served users to retrieve information are browsing and searching. However, browsing does not scale up although it has value in information serendipity. And selective search becomes critical because formulating queries to retrieve the desired information is difficult and requires considerable skill; for example, the use of search engines in the WWW is not standardized. Neither approach appears to be entirely adequate for accessing the wealth of information becoming available. New strategies are needed to deal with information spaces such as the WWW: 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.

For many computer systems, there is a dichotomy between "useful" and "usable." (We are in the process of exploring this difference in the context of high-functionality computer systems [Fischer & Thomas 1993].) Useful systems can be characterized by having a broad functionality and offering a large amount of interesting and relevant information. Usable systems are systems that users can learn and use with a reasonable effort. Usability is easier to achieve for small systems (and many techniques, such as browsing, work reasonably well) than for large complex environments. Adequate tools and support mechanisms are required to achieve the two goals simultaneously. The WWW is a very interesting environment to address this challenge.

In our work we have used agents and usage profiles to increase the usefulness and usability of the WWW. The agents filter information, initiate communication, monitor events, and perform tasks. The agents rely on the usage profiles to adapt their assistance to specific users. In this paper we describe the unique challenges of a WWW-type information space, present the design rationale and conceptual framework behind our work, and illustrate with scenarios the use of the system.

2. Challenges of the WWW

In the WWW, providing information and consuming it are iterative and dynamic processes. We define these processes as the server cycle (for providing information) and the client cycle (for consuming information) of the *WWW Information Lifecycle* (Figure 1), an adaptation of the Information Lifecycle [Fischer & Stevens 1991] to the WWW.

The *server cycle* describes the tasks that have to be done to create a server (or parts of it) for a public information space. Two basic stages can be identified: the production of information and its storage on a server site. Production of information is the process of creating an artifact. This is usually a complex design issue, for example, designing the multi-media layout (called title) for a company's homepage. The information may be customized due to restricted access of the information or due to an information adaptation

specific to different groups (e.g. in-house vs. remote usage). The storage stage is more a technical issue.

The *client cycle* describes the tasks that have to be done by a user to create personal information space as a user-specific view of the global information space. Two basic stages can be identified in this cycle, too: the use of information and the storage of relevant information (links). The use of information is more an interaction issue (browsing or searching), whereas storage is again a design issue. Information can be customized, for example, by adding annotations, downloading, and extracting documents or parts of it.

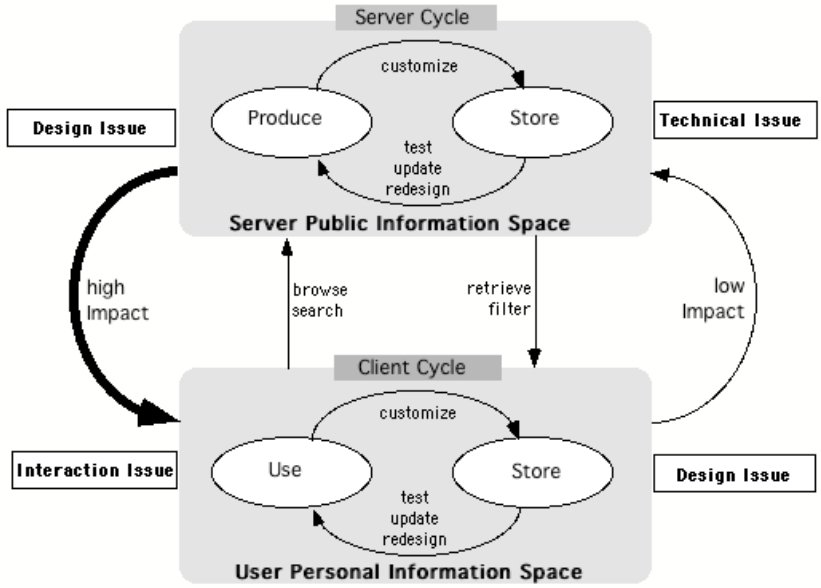


Figure 1: The WWW Information Lifecycle

Changes in the public information space have a high impact on the personal information space, e.g. it may cause problems of information inconsistency in the client cycle. And making a personal view public, for example by creating a personal page with a set of relevant links, has an influence to the public space, too, but with much lower impact. However, the personal page becomes part of the global space.

One typical way of customizing [Berleant & Berghel 1994] the whole information space specific to a user is to create a personal information space or *personal view*. Users build their personal view by (a) their personal assumptions about the structure and content of the WWW, (b) their experiences when browsing and searching, and (c) the kind of information they seek. At first, the system is unaware of this view, which is dynamic because users change their assumptions, experiences, and interests. The view becomes (partly) explicit for the system by the ongoing use of the WWW (selected links are stored in a global history) and by adding personal user interests as bookmarks to a bookmark list (NETSCAPE), as entries to a hotlist (MOSAIC), or by keeping a personal page with a collection of relevant links.

Server cycle and client cycle behave dynamically: information at the public site gets updated, changed, or removed; and at the client site, the user's personal view onto that public space is also changing over time. Links are removed, shifted to other places, or added dynamically.

Discussing the embedding of active software components or *agents* to support the WWW Information Lifecycle is an issue in two directions: (a) supporting the server cycle, and (b) supporting the client cycle. Embedding agents in the server cycle serves the goal of reducing the cognitive load on information providers through active behavior and improving the quality of the designed artifacts [Fischer et al. 1993]. Agents could, for instance, help providers check a company's layout guidelines or provide them with critics of the effects of using different colors in a multi-media title [Nakakoji et al. 1995]. Embedding agents in the client cycle serves the goal to improve the usability and the usefulness of the WWW for information consumers. 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. This paper deals with our work in the second direction.

BASAR (Building Agents Supporting Adaptive Retrieval) illustrates our conceptual framework. BASAR embeds software agents acting as personal assistants in the WWW to improve its usability and usefulness. 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.

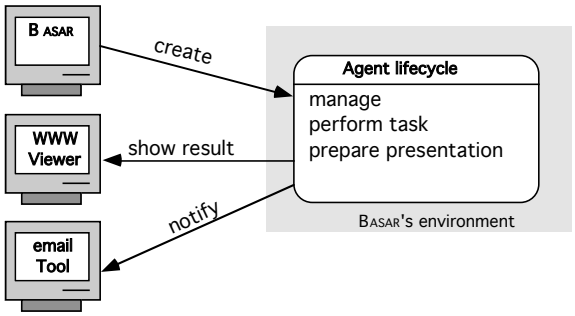


Figure 2: Interfaces of BASAR

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 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. BASAR embodies the following characteristics [Thomas 1995].

- **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 2). This is based on the general idea of intelligent access to large information spaces by converting static information (or knowledge) into an active information (or knowledge) agent [Fischer & Redmiles 1993]. 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, Lemke & Schwab 1985, Krogsæter, Oppermann & Thomas 1994].

- **Active Views**: BASAR provides users with a client (e.g. NETSCAPE, MOSAIC)-independent concept for creating personal information spaces along a semantical meaning. An active view is defined as a set of information links belonging to the same subject—which is a personal view—together with a set of agents attached to that view. 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.

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. The Conceptual Framework

3.1 Indirect Information Management with Agents

We have adopted *indirect management* [Kay 1990] 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.

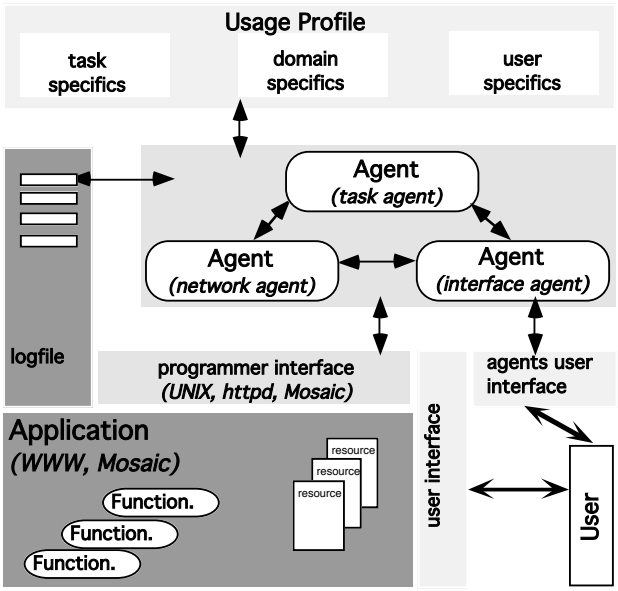


Figure 3: Conceptual Framework

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.

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 an existing application (Figure 3). For the application system, these agents are "add-ons," that is it works with or without agents. Adding new agents complements both the user's interaction with the application and application's interaction with the user.

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 (see Section 3.2).

An agent should be able to act/react, to reason, to perform a task, and to communicate/ cooperate. We make these capabilities be a part of an agent by inheriting the capabilities of four basic objects to an agent: a process, an inference engine, a task performer, and a communicator (see Figure 4 for the complete BASAR taxonomy).

- **Process**: The process capability enables an agent to act or react on a specific situation within an application program. A process knows when it should be active or passive. A trigger mechanism activates the process; once it is activated the process performs the requested tasks from its to-do list. A process trigger for an agent is either defined by a

time description (e.g. ”every hour”) , by an action description (e.g. ”on user request”), or, more generally, by a constraint (e.g. ”until success”).

- **Task performer**: The capability of a task performer enables an agent to support the user within an application domain. The task performer has knowledge about tasks: what are typical tasks and how and when to perform them, for example calling a search engine, looking for updated pages, creating a user-specific WWW document.

- **Inference engine**: The capability of an inference engine enables an agent to infer knowledge from a knowledge base. Using rules, an inference engine updates the facts in a knowledge base relevant to the actual context. A typical knowledge base in our system is the usage profile.

- **Communicator**: The capability of a communicator enables an agent to communicate and exchange data and knowledge with the user, with the application, and with other agents.

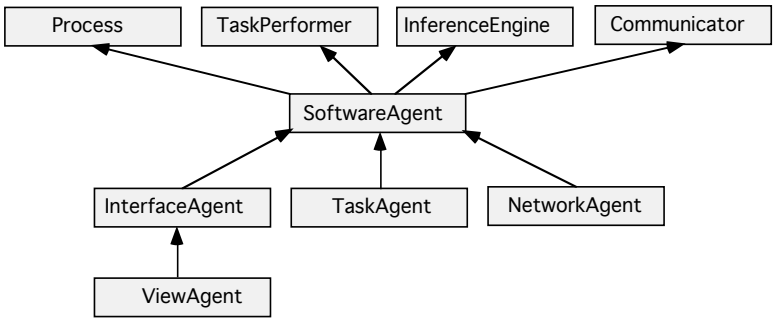


Figure 4: BASAR Agent Taxonomy

The arrows represent the ”is-a” relation. If a user defines a new agent, then a new instance of the class Task Agent is created. Also, the predefined agents that come with BASAR (see Table 1) are instances of that class.

Together, all four basic object capabilities enable an agent to act like an assistant to the user by performing a task within an application domain (semi-) autonomously, when it is requested, specific to the user’s needs and constraints inferred from the usage profile.

For the purpose of BASAR, we have created three subclasses of the class Software 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), as shown in Figure 4.

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 1.

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.

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 1: Built-in Task Agents

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 7), to edit the usage profile (see Figure 5), 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.

3.2 Usage Profiles

The usage profile is the central knowledge base in our framework. It is a set of objects that condenses the knowledge about the tasks, the domain, and the user. Rules are added to infer context-sensitive support [Thomas 1993]. Acquiring knowledge for the usage profile, identifying the relevant objects, and representing them in the knowledge base is an iterative process—design, evaluation, redesign—based on user-specific needs.

For example, tests we conducted with the browser MOSAIC and the WWW confirmed that users have difficulties in making efficient use of search engines (see scenario, Section 4). In consequence, we identified the need for an agent that specifically supports that task. The question was how to fill the usage profile with search-relevant attributes and build a domain-specific knowledge base about search engines.

At first, we used our own experiences with searching in the WWW for a design of that search agent. 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.

As a next step, we are currently validating these results with empirical tests. The tests are done by analyzing a questionnaire about the usage of search engines; that questionnaire has been distributed in some GMD internal newsgroups.

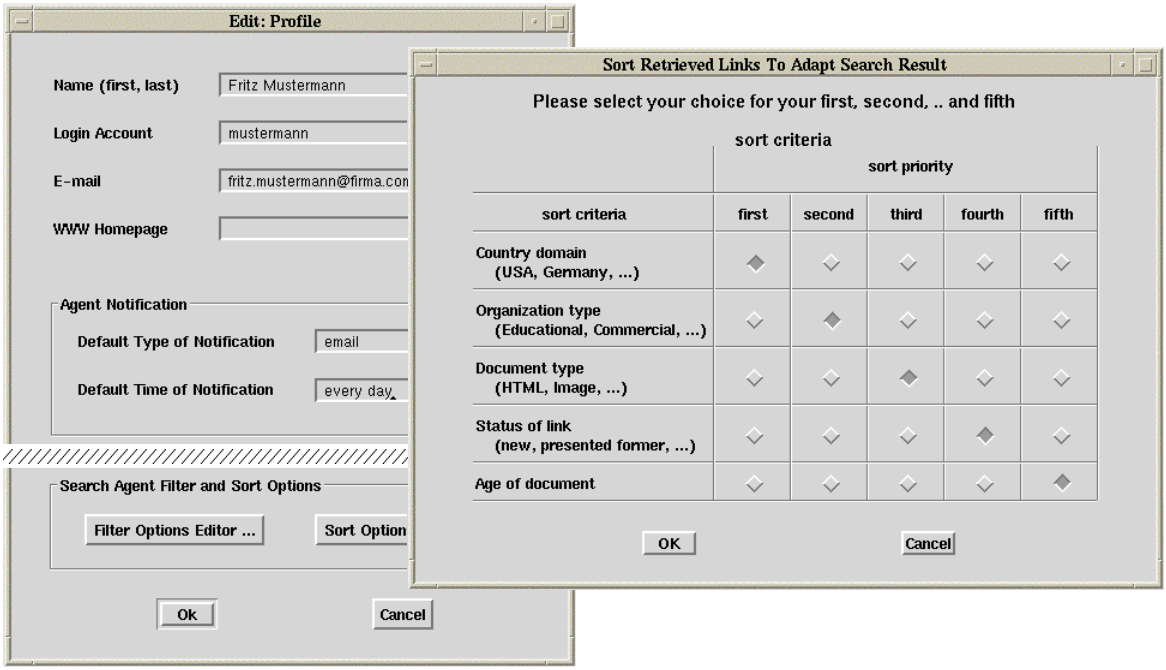


Figure 5: Editing the Usage Profile

The top-level window of the usage profile contains user's initial data such as name, account, email, preferred way and time of getting notified by agents, for example by email once a day, see left. In the lower part of that window, user has access to the user-specific attributes concerning filter and sort options for search agents, for example.

The window on the right pops up after selecting the button "sort options editor". It presents a table of user's preferences how to sort the search results. The selected values in the table have been infered by looking at user's actions on former search results.

Our previous work on malleable [Fischer 1993] and adaptive systems showed that there is a strong demand for complementing implicit modeling (observing the user) with explicit modeling techniques (asking the user) and vice versa. Therefore, in BASAR we use both techniques to build, then refine, a dynamic model [Mastaglio 1991].

When a new user starts working with BASAR, the first modeling usually is done explicitly, for example, by a dialog box that pops up and asks the user for some user-specific data such as name, email address, and preferred way to get notification from the agent. These initial data are used for the agent's user interface.

The next step in usage modeling is implicit: all actions the user is doing in WWW and BASAR (e.g. creating a new agent, modifying an existing one, or creating an active view) are analyzed and the results are stored in the usage profile. For example, when the user creates a search agent with a key word such as lifelong learning (see the scenario below),

all relevant information is stored in the profile: the name of the agent, the keyword, when created, the returned results, and what the user did with the results (having a look at, ignoring, already stored in an active view, etc.).

Our experiences [Oppermann 1994] have shown that users want to know how their own system usage is "modeled." This requires that (parts of) the usage profile should be open to the user, such as displaying relevant attributes about the user and the system usage [Hill et al. 1992] and making them editable through a usage profile editor (Figure 5).

4. 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.

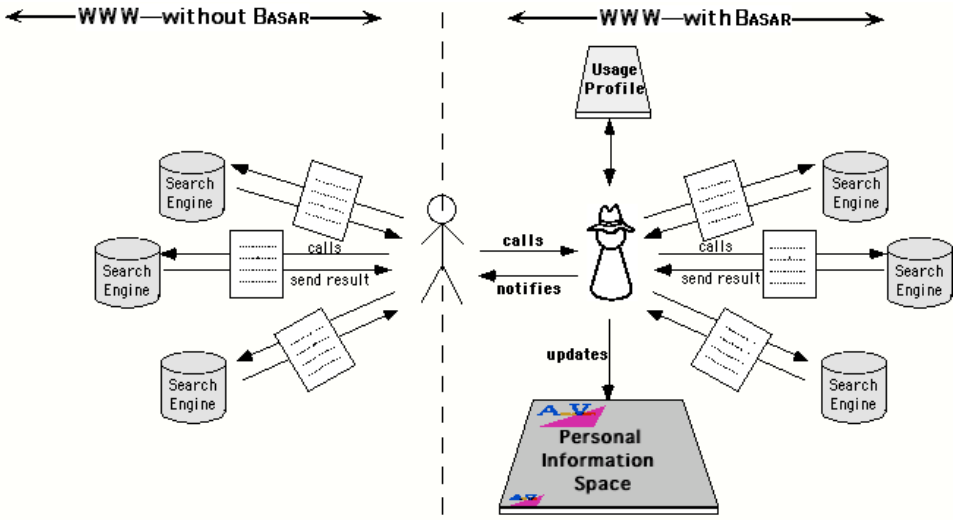


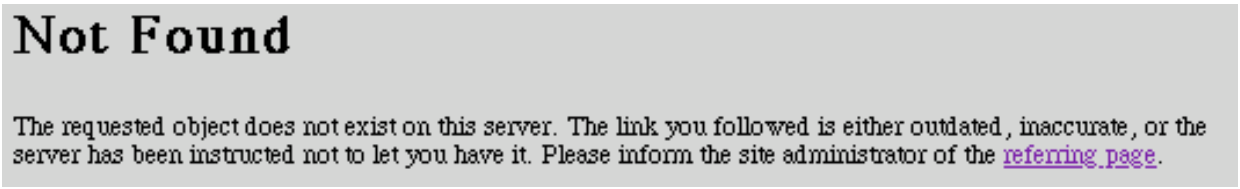
Figure 6: Searching in the World-Wide Web without and with BASAR

WWW—without BASAR. One usual way to start with the WWW is to call one (or more) search engines that are requested to return links relevant to the given search key. It is then the user's task to filter the result (Table 2) by hand (Figure 6, left part).

The user's time and experience determine how the user will interpret the results and take the relevant links for building the personal information space concerned with "lifelong learning." But the problems 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 (these are results of the tests we conducted, as reported in the former section on usage profiles).

- **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:



- **Multiplicity**: Users needs 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 2: 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. Once an initial personal space for "lifelong learning" has been created, the next problem is that this view, according to the WWW Information Lifecycle model, is dynamic because it changes over time. What is missing is

- an active support to inform the user about ongoing changes and updates in the personal information space, and
- a functionality to classify bookmarks or hotlist entries according to their meaning—which may be done either explicitly (by the user) or implicitly (by the system).

To overcome these problems, we support the search process (WWW with BASAR) by introducing agents, usage profiles, and active views.

WWW—with BASAR. 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 7). 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 1). The search agent mediates between different search engines and the user (Figure 6, right part). Its behavior is influenced by the information contained either in the logfile or in the usage profile.

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.

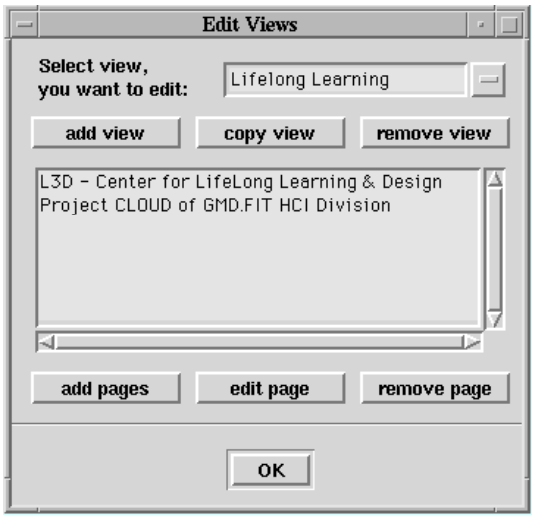


Figure 7: Creating the Active View "Lifelong Learning"

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.

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.

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.

5. Related Work

Software agents technology is a growing area in different research fields: in CSCW (Computer-Supported Cooperative Work) with the focus on cooperation, in HCI (Human-Computer Interaction) with the focus on assistance, and in AI (Artificial Intelligence) with the focus on distributed problem-solving. Moreover, agent 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 & Warner 1994]. 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 & Hawkins 1994], or TELESCRIPT lets users send executable programs in the form of agents through the network [Wayner 1995].

In the WWW, search engines, also called Internet Agents [Indermaur 1995], 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 and Ehrlich 1995]. For example, HOMR [Shardanand & Maes 1995] is a collaborative information-filtering system based on learning agent technology [Maes 1994].

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 two earlier implemented systems, FLEXCEL [Thomas 1993], as an adaptive user interface extension for the spreadsheet program Excel® from Microsoft, and INFOSCOPE [Fischer & Stevens 1991, Stevens 1993], as an information-filtering system that uses a simple agent scheme to assist users in managing the large information space of USENET news.

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.

8. 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 and information providers.

One of the shortcomings of the current version of BASAR is that it does not do any content analysis of the information. Therefore, we are currently discussing the idea of using the Latent Semantic Indexing (LSI) algorithm [Deerwester et al. 1990] for the purpose of BASAR. We are testing whether an LSI content analysis of an active view can be used as an additional source in the usage profile for predicting which information links of a search result may be most useful to extend that view.

Another 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 our research groups at CU and at GMD as a joint information repositorium. This idea follows the discussions of collaborative work practices [Nardi 1993], which imply that not all users necessarily have to create their own agents, but we will have power users (the ones that create the view and the agents on that view) and local users (the ones that can add new information links by using a bbs system and periodically get informed about the common information space). Concerning user modeling techniques, this raises new issues of matching and complementing usage profiles of single users with usage profiles of groups.

The other extension of BASAR has been briefly mentioned when introducing the WWW Information Lifecycle: using the concepts of agents to support information providers or authors in WWW. Information providers are dealing with a large amount of information. For example, the WWW server of GMD contains hundreds of documents (pages for institutes, research groups, projects, people, and articles) and thousands of objects (text,

picture, video, audio). Agents integrated as assistants on the server cycle could give feedback to the "webmasters" and the "web-editors" about the ongoing status of that dynamic space, making its management more convenient than it is presently.

Acknowledgments
The authors would like to thank the members of the Center for LifeLong Learning and Design at the University of Colorado, who have contributed to the conceptual framework and systems described in this paper. The research was supported in part by (1) NSF Grant IRI-9311839, "Enhancing Indirect, Long-Term Collaboration with Intelligent Agents," (2) the ARPA HCI program, Grant N66001-94-C-6038, (3) Nynex, Science and Technology Center, (4) Software Research Associates (SRA), and (5) PFU. We would also like to thank Martin Kohnen, who is working on the user modeling part of BASAR, and Achim Nick, who did most of the implementation of BASAR, both at GMD.

References

[Berleant & Berghel 1994]
Berleant, Dan / Hal Berghel: *Customizing Information: Part 1: Getting what we need, when we need it*. IEEE Computer, Vol. 27, No. 9, September 1994, pp. 96-98.

[Deerwester et al. 1990]
Deerwester, Scott / Susan T. Dumais / George W. Furnas / Thomas K. Landauer / Richard Harshman: *Indexing by Latent Semantic Analysis*. Journal of the American Society for Information Science. Vol 41, No. 6, 1990, pp. 391-407.

[Fischer 1993]
Fischer, Gerhard: *Shared Knowledge in Cooperative Problem-Solving Systems - Integrating Adaptive and Adaptable Components*. M. Schneider-Hufschmidt, T. Kuehme, U. Malinowski (Eds): Adaptive User Interfaces - Principles and Practice, Elsevier Science Publishers, Amsterdam, 1993, pp. 49-68.

[Fischer & Thomas 1993]
Fischer, Gerhard / John Thomas: *Designing Useful and Usable Computational Environments*, ARPA-HCI Grant N66001-94-C-6038, Boulder, CO, 1993

[Fischer et al. 1993]
Fischer, Gerhard / Kumiyo Nakakoji / Jonathan Ostwald / Tamara Sumner: *Embedding Critics in Design Environments*. In: The Knowledge Engineering Review Journal, Cambridge University Press, No. 4, Vol. 8, December 1993, pp. 285-307.

[Fischer, Lemke & Schwab 1985]
Fischer, Gerhard / Andreas C. Lemke / Thomas Schwab: *Knowledge-Based Help Systems*. Human Factors in Computing Systems, CHI'85 Conference Proceedings (San Francisco, CA), ACM, New York, April 1985, pp. 161-167.

[Fischer & Redmiles 1993]
Fischer, Gerhard / David Redmiles: *Enhancing Indirect, Long-Term Collaboration with Intelligent Agents*. NSF-Grant IRI-9311839, Boulder, CO, 1993.

[Fischer & Stevens 1991]
Fischer, Gerhard / Curt Stevens: *Information Access in Complex, Poorly Structured Information Spaces*. Human Factors in Computing Systems, CHI'91 Conference Proceedings, ACM, New York, 1991, pp. 63-70.

[Guilfoyle & Warner 1994]
Guilfoyle, Christine / Ellie Warner: *Intelligent Agents: the New Revolution in Software*. Ovum Limited, London, UK, May 1994.

[Hill et al. 1992]
Hill, W.C. / J.C. Hollan / D. Wroblewski / T. McCandless: *Edit Wear and Read Wear*. Human Factors in Computing Systems, CHI'92 Conference Proceedings (Monterey, CA), ACM, May 1992, pp. 3-9.

[Indermaur 1995]
Indermaur, Kurt: *Baby Steps*. BYTE, March 1995, pp. 97-104.

[Kay 1990]
Kay, Alan: User Interface: A Personal View. Brenda Laurel (Ed): *The Art of Human-Computer Interface Design*, Addison Wesley Publishing Company, Inc., 1990.

[Krogsæter, Oppermann & Thomas 1994]
Krogsæter, Mette / Reinhard Oppermann / Christoph G. Thomas: *A User Interface Integrating Adaptability and Adaptivity*. R. Oppermann (Ed.): Adaptive User Support. Lawrence Erlbaum Associates 1994, pp. 97-125.

[Maes 1994]
Maes, Pattie: *Agents that Reduce Work and Information Overload*. CACM, July 1994, Vol. 37, No. 7, pp. 31- 40.

[Maltz & Ehrlich 1995]
Maltz, David / Kate Ehrlich: *Pointing The Way: Active Collaborative Filtering*. Mosaic of Creativity, CHI'95 Conference Proceedings, ACM, New York, 1995, pp. 202-209.

[Mastaglio 1991]
Mastaglio, Thomas: *A User-Modelling Approach to Cooperative Problem Solving*. Ph.D. Thesis, University of Colorado, 1991.

[Nakakoji et al. 1995]
Nakakoji, Kumiyo / Brent N. Reeves / Atusi Aoki / Hironobu Suzuki / Kazunori Mizushima: *eMMaC: Knowledge-Based Color Critiquing Support for Novice Multimedia Authors*. Proceedings of ACM Multimedia'95, ACM Press, San Francisco, CA, forthcoming, 1995.

[Nardi 1993]
Nardi, Bonnie A.: *A Small Matter of Programming. Perspectives on End User Computing*. The MIT Press, Cambridge, Massachusetts, London, England, 1993.

[Oppermann 1994]
Oppermann, Reinhard (ed.): *Adaptive User Support*. Lawrence Earlbaum Associates, Publishers, Hillsdale, New Jersey, 1994.

[Roesler & Hawkins 1994]
Roesler, Martina / Donald T. Hawkins: *Intelligent Agents - Software Servants For An Electronic Information World (and More!)*. ONLINE, Volume 18, No. 4, July 1994, pp. 19 - 32.

[Schick, Gordon & Haka 1990]
Schick, Allen G. / Lawrence A. Gordon / Susan Haka: *Information Overload: A Temporal Approach*. Accounting Organizations and Society, Vol. 15, No. 3, 1990, pp. 199-220.

[Shardanand & Maes 1995]
Shardanand, Upendra / Pattie Maes: *Social Information Filtering: Algorithms for Automating "Word of Mouth"*. Mosaic of Creativity, CHI'95 Conference Proceedings, ACM, New York, 1995, pp. 210 - 217.

[Stevens 1993]
Stevens, Curt: *Knowledge-Based Assistance for Accessing Large, Poorly Structured Information Spaces*. Dissertation, University of Colorado at Boulder, CU-CS-640-93, February 1993.

[Thomas 1993]
Thomas, Christoph G.: *Design, Implementation, and Evaluation of an Adaptive User Interface*. Knowledge-Based Systems. Special Issue on Intelligent Interfaces. Vol. 6, No. 4, December 1993, pp. 230-238.

[Thomas 1995]
Thomas, Christoph G.: *Basar: A framework for integrating agents in the WorldWide Web*. IEEE Computer, Vol. 28, No. 5, May 1995, pp. 84-86.

[Wayner 1995]
Wayner, Peter: *Free Agents*. BYTE, March 1995, pp. 105-114.