A Multiagent system for Navigation Assistance and Information Finding

Samhaa El-Beltagy, David De Roure and Wendy Hall

Multimedia Research Group
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
+44 (0)1703 592418
{seb,dder,wh}@ecs.soton.ac.uk

Abstract

A multiagent framework has been developed to address distributed multimedia information management. Our particular interest is information management within organisations, where information is a vital resource and locating the right information at the right time is a critical problem. Hence our focus is navigation assistance, and we are pursuing the idea that information acquired by one user navigating the information space is something that can be utilised to assist other users. The design of the system we present builds on the experiences of an earlier project which provided navigation assistance based on information about which documents other users had visited; our system has an improved architecture and extends the functionality, focusing on bookmarks instead of 'trails'. We have also integrated additional services including handling of multimedia documents and integration with other open hypermedia systems. The system is being evaluated within commercial organisations.

1. INTRODUCTION

The problem of finding information on the Web is well known. The Web is the largest distributed hypermedia systems known to date, and certainly, in terms of users, the most used. In 1997, estimates suggested that the Web included 150 million documents with a doubling rate of approximately four months [11]. Unlike most hypermedia systems however, the Web lacks the conceptual model which organises and maintains data and link consistency. Thus, despite the abundance of Web search engines, trying to locate information on the Web is still analogous to locating a needle in a haystack. The fact that information systems that are both distributed and heterogeneous in nature continue to become available on the Web complicates the problem even more. Consequently, attempting to locate, integrate and organise related information has become a major challenge. Terms such as "information overload" are becoming commonplace and huge research activities are being carried out in an attempt to tame the Web and the information contained within it.

In [6] we discussed the role of agents for distributed information management, which included resource discovery, information integrity and navigation assistance. Our approach is based on the principles of open hypermedia, where link information is managed separately from documents, and includes the ability to obtain available links based on features extracted from multimedia document content; i.e. we integrate hypermedia with a more query-oriented approach. We presented a case study in navigation assistance in [21], focusing on a project called MEMOIR, and an approach to hypermedia link integrity was explored in [18]. These and other projects are part of our rich information management environment, much of which is based on open hypermedia.

In this paper we introduce a collaborative multiagent system that has been developed in an attempt to integrate some of the services developed at the University of Southampton that can assist users working within research groups or organisations, in locating information they need either within their local information systems or on the Web, as well as to innovate some new ones that serve the same purpose. Within the presented architecture, each agent performs a relatively simple task, but collectively, they present the users of the system with a powerful set of tools that help in finding and creating information. The system also assists the users in Web navigation and in finding other users within an organisation or group who share similar interests. The goals of the proposed system can be summarised as follows:

- Making maximum use of what users do normally while browsing/searching the Web. Such activities include viewing and bookmarking various documents. Bookmarks provide a natural way for organising documents into related categories. By enabling users to make some of their bookmarking activities public, users within organisations or groups are capable of locating related documents from high quality stores of information. By comparing bookmarks, matching of human interests is also possible.
- Allowing users to augment documents or aspects of documents, with useful data. This could be a rating for the document, a creation of a generic link (explained in section 3.3) to the document, or a creation of a link from the document to another, etc.
- Allowing search activities to be carried out through multiple information stores, transparently.

This paper is organised as follows: Section 2 provides a general background for the work presented, section 3 describes the architecture and each of the agents that operate within it, section 4 discusses related work, section 5 reviews some features that we would like to implement in the future, and finally, section 6 summarises and concludes this paper.

2. BACKGROUND

The system presented in this paper builds on our experience in building the MEMOIR project which adopted an agent based architecture [7][21]. One of the main goals of the MEMOIR framework was to support researchers working with vast quantities of distributed information in finding both relevant documents and researchers with related interests. MEMOIR employed an open architecture based on the existing Web infrastructure and used proxies as key components. The system can be described as a collaborative recommendation system where users can group a set of documents they perceive as interesting into trails that are stored in a shared organisation memory. Using this knowledge, the system can recommend documents to other users based on what they find interesting by attempting to locate relevant documents in trails. More importantly, the system uses knowledge about what its users find interesting to provide pointers to users that share similar interests.

In MEMOIR, trails were key to some important system functions. They were used to match user interests as well as to perform document recommendation services. A trail in MEMOIR is defined as a collection of URLs that a user creates during a browsing session and which contains documents that are related to one another and which the user has found interesting. Trails are created through non-persistent agents which send the trail to a data store where it is kept for future reference. For a single user, bookmarks are a powerful facility that can be used to reference documents that a user has found useful and which he/she wants to be able to get back to. A collection of bookmarks under a given category also denotes a relationship between those bookmarks as indicated by the category. In a sense, a given bookmark category could be thought of as a trail that has evolved over time, rather than from a single browsing session. Bookmarking is a very natural activity for most Web users and unlike the creation of trails that are never directly accessible to them again, does not impose an overhead in terms of effort. For that reason, bookmarks have replaced trails in this system.

A number of agents presented in this work have been influenced by the Microcosm [5] and Distributed Link Service[3] systems, also developed at University of Southampton. The way in which these systems influenced each agent is detailed in section 3 where relevant.

3. SYSTEM OVERVIEW AND ARCHITECTURE

The system to be described in the following sections is a collaborative multiagent one. Multiagent systems have been identified as essential to the successful engineering of complex or large systems. A multiagent system is composed of a group of agents that are autonomous or semiautonomous and which interact or work together, to perform some tasks or achieve some goals. The agents in such systems may either be homogeneous or heterogeneous and they may have common goals or goals that are distinct [16]. The design of individual agents within a multiagent system has the advantage of being independent of the design of other agents as long as each agent abides by an agreed upon protocol and ontology. This significantly contributes to the breakdown of complexity. Agents are thus viewed as black boxes whose operations are abstracted to the services they provide, and which they announce to a facilitator agent. Agents in the presented system can be viewed as information producers and consumers. For example, a user interface agent acquires knowledge about what different users are viewing and with the permission of the user, shares that information with others, through another agent. It then uses the services offered by agents that utilise this information, to find out about other users' interests or to obtain a recommendation for a document, among other things.

As one of our objectives is to investigate agent frameworks for distributed information management in the context of open hypermedia, we have constructed our own infrastructure components. Messages between agents are exchanged in KQML [9][15] as the outer language and Prolog as the inner language. Communication is carried out over TCP/IP sockets. Most of the agents are implemented in Java, with the exception of one agent, which is implemented in Prolog. Services that directly benefit a user are made available through the user interface agent. The user interface agent in turn knows about the available services and their provider agents through a facilitator agent.

Within the presented framework, each user is associated with a personal interface agent that acts on his/her behalf. In addition, a number of agents that support the user in various tasks exist within this framework. The agents are distributed across different machines, and either act independently or collaborate with other agents to carry out their problem solving tasks. In the following subsections, we describe each of the agents in detail. Figure 1 shows a simplified diagram of the presented architecture. Messages are exchanged based on an asynchronous communication model.

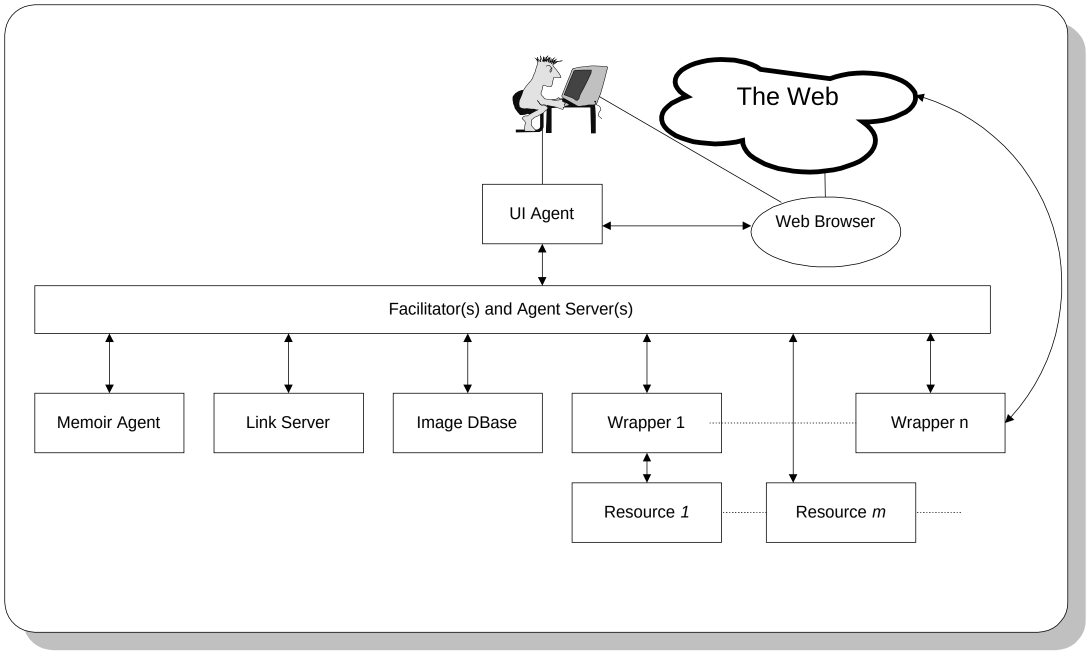

Figure 1: A simplified diagram of the proposed architecture

3.1 The Facilitator

One of the most important agents within this framework is the facilitator agent (the agent server) that contributes to the openness and extensibility of the presented architecture. A facilitator agent within this architecture complies with notions of facilitation as defined by Khedro and Genesereth [13]. Mainly, it supports routing of information, discovery of information, delayed and persistent notification of messages and communication management. When agents come online for the first time, they register with the facilitator agent using the KQML register performative. They then advertise their services to the facilitator, which in turn broadcasts the advertisement to all registered agents. As soon as an agent registers, it also receives all active advertise messages that were previously sent to it by other registered agents.

Implementation

The facilitator agent has been implemented in Java. A message to the facilitator agent goes through several stages before being processed. One of the components of the facilitator agent is the message listener, which is a multithreaded Java server built on top of sockets. Once a message is received, it is passed on to a KQML message verifier. This component employs a Java KQML library that has been developed and reused across most of the implemented agents. The KQML library takes as input the string KQML message and converts it into a KQML message object that allows an application to inquire about the validity of the message, as well as the value of various fields of the message examples of which are the sender, the receiver, the content, etc. The KQML message verifier checks if the message is valid. If it is not, the appropriate error message is dispatched to the sender. It then checks to see if the message is addressed to an agent other than itself. If that is the case, it forwards the message to the agent if it is known, otherwise, it replies with an appropriate message. If the receiver of the message is not specified, then it tries to match the service requested in the content of the message with services advertised by other agents and forwards the message to the appropriate agent accordingly. If the message is addressed to itself, the performative of the message is checked, and a method for that particular performative is activated. If the performative is not supported, a 'sorry' message is dispatched.

Delayed and persistent notification is achieved through the use of a messaging thread. The messaging thread contains a message queue to which a message is appended upon failure of delivery. The thread sleeps for a given amount of time, and when it wakes up, it attempts to deliver all messages in its queue, and then goes to sleep again. Each message is associated with a counter and when message delivery fails for *n* times, the message is erased from the queue.

3.2 The Organisational Memory agent (OM agent)

This agent serves as an organisational memory for the browsing and bookmarking activities of users within the system. Like all agents within this framework, when this agent comes on line for the first time, it registers with a facilitator and then starts advertising its services. The agent allows other agents to register users and then to publish information about their activities. Basically, the agent can be told that a given user has seen or bookmarked a given URL. A bookmark is defined in terms of a URL, a title, a category, and a set of keywords. It can also be told how a user rated a given URL. This information is used in a number of ways. By knowing who has seen what, the agent is capable of answering a question such as “*Who else has seen the following URL?*”. Bookmarks are used in more intelligent way to inference knowledge that is not as straight forward. Other agents can still ask “*Who has bookmarked the following URL?*”, but based on bookmarks, they can also ask this agent to respond to queries such as “*Recommend URLS related to this*”, “*Recommend URLs related to this emphasising the following keywords*”, “*Who is interested in a given topic?*”, and “*Find all documents that best satisfy the following keywords*”.

The algorithms used to inference the answers to these queries are similar, though not identical to those employed by MEMOIR [21]. To recommend URLS similar to a given document, all URLs in categories for all users in which this URL appears, are fetched. Ordering of the returned results is based on the duplication of each URL in other categories. None of the URLs are discarded. In case emphasis is put on some keywords, they serve as additional constraints in the inferencing process. To answer who is interested in a given topic, two steps are followed. The first is a straightforward approach where the user interests provided in registering a user, are searched for the topic. It is assumed that the user would have probably supplied this information. The second step involves the consideration of the topic as a keyword and searching the bookmarks for occurrences where this keyword appears for a number of times that is higher than a given threshold. The results all then ranked according to maximum occurrence. Finding documents based on keywords, is achieved by simply the searching through the bookmarks for the URLS that satisfy the most of the given keywords.

The following KQML message, shows one of the advertise messages sent to the facilitator. In this example, the OM agent is advertising its capability to be told that a user denoted by a given email address has seen the given URL.

```
(advertise
        :sender memoir.snoopy
        :receiver AgentServer
        :reply-with ""
        :language KQML
        :ontology kqml-ontology
        :content (tell
                        :sender “”
                        :receiver memoir.snoopy
                        :reply-with id7889
                        :language Prolog
                        :ontology memoir -ontology
                        :content seen (+URL, +Email) ) )
```

Other tell messages that the agent is capable of receiving include:

- person(+Email, +FirstName, +LastName, +URL, +Interests, +Phone, +Fax)
- bookMarked(+Keywords, +Title, +Category, +Url , +Email)
- rating(+URL, +Email, +Rating)

The agent also advertises it capability to respond to the KQML ask-all performative with the following content:

- whoElseHasSeenThis(+URL, +Email, -PersonList)
- whoElseHasBookMarkedThis(+URL, +Email, -PersonList)
- recommendSimilarDocs(+URL, +Email, -UrlList)
- recommendSimilarDocs(+URL, +Keywords, +Email, -UrlList)
- whoIsIntrestedInThis(+topic, +Email, -PersonList)
- find(+Keywords,'docs', - UrlList)

The agent can only respond to an *ask* performative with the following content:

- getRatingFor(+URL, -AverageRating, -NoOfVotes)

Implementation

The knowledge base of this agent has been implemented in Prolog. Prolog has been chosen because it easily supports the required functionality and inferencing capability. The interface of OM agent with other agents has been implemented in Java. The Java component validates any incoming messages before passing their content on to the Prolog component.

The Prolog Knowledge Base

This agent advertises its ability to be told about browsing and bookmarking activities. In KQML terms, it advertises that other agents could insert facts into its virtual knowledge base. In reality, other agent can assert those facts into its real knowledge base. The following facts are examples of such assertions.

- `seen('http://www.mmrg.ecs.soton.ac.uk', wh).`
- `person(wh, 'Wendy','Hall','http://www.ecs.soton.ac.uk/~wh', ['Multimedia', 'Open hypermedia', 'Hypertext'], '4060', '9090').`
- `bookmarked(['agents', 'software agents', 'intelligent agents', 'intelligent software agents', 'kqml', 'kif', 'knowledge sharing', 'mobile', 'softbot', 'fipa', 'acl', 'ontology'],'Software Agents', 'UMBC Agent Web', 'http://www.cs.umbc.edu/agents/', dder).`

These facts are asserted in a persistent data store to ensure that the information they represent will not be lost in case of failure of any sort. Prolog methods by which queries could be answered and inferenced are also part of the knowledge base. The following examples show the Prolog code for answering two of the declared queries:

```
whoElseHasSeenThis(URL, Current, Users):-db_fetch(seen(All, URL), _),
                                          delete(All,Current,U),
                                          process(U,Users).

recommendSimilarDocs(URL, Related) :-
          db_fetch(bookmark(_,_, C,URL,_),_),
          findall(Bookmrk,         ((db_fetch(bookmark(_,_,C,Bookmrk,_),_),
          not(Bookmrk=URL))),Result),
          order(Result, O_Results)
```

```
            remove_duplicates(O_Results,Related),!.
recommendSimilarDocs (_,[]).
```

3.3 The Link Service Agent

The idea of this agent is inspired by the Distributed Link Service(DLS) System[3] that was in turn inspired by ideas presented in Microcosm[5]. Microcosm is an open hypermedia system which allows links to be stored separately from documents. The concept of abstracting links from documents allows for a great deal of flexibility since it means adding hypermedia functionality to documents without changing the document format or embedding mark-up information. One type of a link introduced in Microcosm, is the generic link. In the context of text documents, generic links allow a link to be followed from any occurrence of a text string to a destination anchor. By employing generic links, a great amount of re-authoring is avoided, and link re-use is greatly enhanced. DLS allows links, mostly generic, from linkbases to be applied to WWW documents.

The Link Service Agent, is an agent that provides services that are very much like the DLS, but in addition achieves a number of useful tasks based of the concept of a link being an unresolved query. In the original Microcosm model, a user was offered the capability of computing links related to a word by simply highlighting any word in a document and pressing a button. All links that are relevant to that document were then collected from linkbases and displayed to a user. In the DLS, this feature was difficult to achieve using the Web technology available at the time. Since such a service is now possible through Javascript technology, we have re-introduced this feature again through this agent. The agent thus receives a request for links given a word or a group of words, and searches through its database of user created links, for appropriate matches. It also employs the services of all agents that are capable of answering a KQML message of the type: ask-all, with content: find(+Keywords,'docs', - UrlList), to compute the desired links, thus utilising the abilities of the other agents in query resolution for link computing. The local linkbase itself is constructed through *createLink* type of messages received from other agents. It provides services of the *applyLinks* and *computeLinks* type.

3.4 The Image Annotator Agent

Searching for multimedia information embedded in WWW pages is still an area where much research is being conducted. However, little attention has been directed towards implementing tools for locating and viewing parts/segments of multimedia. Specifically, the limitations of image representation on the Web could be summarised as follows:

1. The representation of multimedia fragments is currently not well supported.
2. There are no means by which authors can specify links to multimedia fragments.
3. Augmenting multimedia fragments, or even entire multimedia elements, with metadata is not supported.

Since one of the main aims of this work is to assist users in finding information multimedia or otherwise, these limitations were addressed [8]. The inspiration for this work lies in Microcosm [5] in which the idea of linking from parts of media elements to text fragments or other media elements and vice versa, is one which is well supported.

In order to address point 1, an image viewer applet capable of displaying an image with a highlighted region was implemented [8]. This tool indirectly supports point 2 since this applet could be embedded in an HTML file that denotes the destination of a link. In the context of information finding, point 3 is perhaps the most important. By augmenting images and image regions with metadata, information that would otherwise be inaccessible is made available. For example, if a map is annotated for regions that represent cities, then searching for a city will not only yield a pointer to the map, but a pointer to the city on that map.

Following the same premise, that users can benefit from each other's experience, an agent has been implemented to capture the user's annotations of image regions assuming that they are provided with a tool that will allow them to create such annotations irrespective of the underlying implementation of such a tool. The image accepts these metadata annotations and utilises them in answering queries. Metadata for an image is defined on two different levels. The image metadata level, and the content metadata level. On the first level, descriptors that are usually defined only once for the entire image, are specified. These include the image URL, the image Title/Description and the image creator. In addition, a simple classification for the images could be defined on that level. Five basic classes for images have been identified: photographs, paintings, maps, and diagrams. However, the system does not restrict the classification of images under any of these categories, but instead allows the dynamic creation of new categories. The second level is defined in terms of a title, keywords, and a comment/description. The KQML messages that this agent advertise as capable of understanding are:

- Performative: tell, ontology: mna, content: imageRegion(+URL, +ImageTitle, +Creator, +RegionTitle, +Keywords, +Category, +x1, +y1, +x2, +y2)
- Performative: ask-all, ontology mna, content: find(+Keywords, +Category, - UrlList)
- Performative: ask-all, ontology mna, content: applyLinks(+HtmlText, 'Links to Images', -ModifiedHtml).

Implementation

This agent has been completely implemented in Java. Upon receiving the imageRegion annotation, it generates an HTML file in a configured directory accessible to a Web Server, to represent that region utilising the developed image viewer applet. It then uses the keyword and category information to create indices to the generated HTML file. Upon receiving a query, it uses those indices to return appropriate image pointers given by URLs of suitable HTML files. Figure 4 shows a diagram of how this agent is used within the described framework.

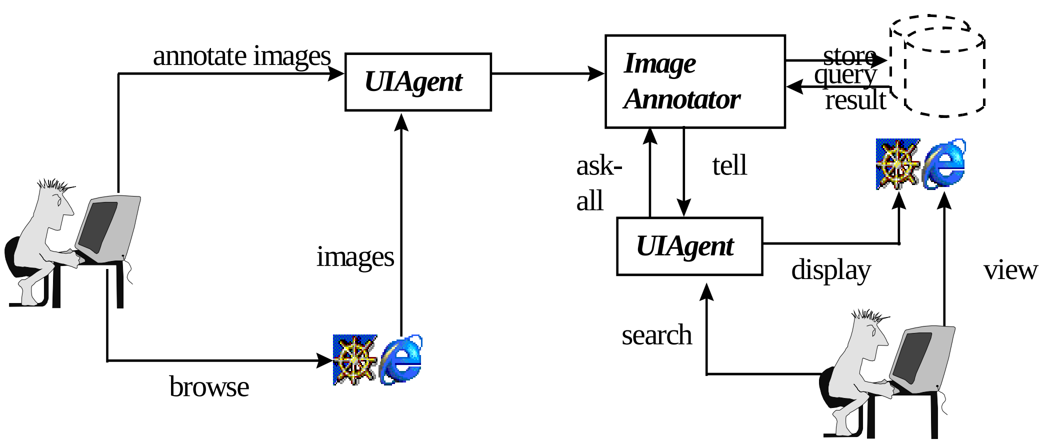

Figure 2: The image annotator as used within the framework

3.5 Search wrappers

As a proof of concept to the flexibility of the framework in dynamically incorporating agents, two wrappers around existing systems were developed. The first of these, is a wrapper around the

Electronics and Computer Science staff, and students database. The second is a wrapper around the Altavista Search engine [1]. Both wrappers accept the KQML ask-all performative, with the ontology: mna and content: find(+Keywords, ’doc’, -UrlList). Both wrappers were implemented in Java and employ multithreaded message listeners.

3.6 The User Interface Agent

Within the proposed framework, each user interface agent is associated with a user. This agent has a number of important functions:

- It provides the user with an interface whereby he/she can make use of the services available in the system through transparent communication with agents that are needed to the accomplishment of a service required by a user.
- It keeps information entered by users about themselves, such as what they consider their interests, their contact information, Web pages, etc.
- It provides other agents within the system with information about the user and his/her browsing and bookmarking activities which can be used by other agents to achieve a number of tasks
- In cases where a user enters a query, the agent is responsible for integrating the responses returned from various agents and presenting them to the user.
- If a service can change the way in which a normal document is presented, the UI agent is responsible for making appropriate changes.

The first time the agent is activated, it registers with the facilitator agent. It then receives all the advertise messages sent by other agents. The agent seeks the services of agents that employ the mna ontology and the Prolog language and which are capable of the receiving the following messages:

1. Performative: tell, content: person(+Email, +FirstName, +LastName, +URL, +Interests, +Phone, +Fax)
2. Performative: tell, content: seen(+URL, +Email)
3. Performative: tell, content: bookMarked(+Keywords, +Title, +Category, +Url , +Email)
4. Performative: tell, content: rating(+URL, +Email, +Rating)
5. Performative :tell, content: imageRegion(+URL, +ImageTitle, +Creator, +RegionTitle, +Keywords, +Category, +x1, +y1, +x2, +y2)Performative: ask-all, content: whoElsehasSeenThis()
6. Performative: ask-all, content: whoElseHasBookMarkedThis(+URL, +Email, -PersonList)
7. Performative: ask-all, content: whoElseHasSeenThis(+URL, +Email, -PersonList)
8. Performative: ask-all, content: recommendSimilarDocs(+URL, +Email, -UrlList)
9. Performative: ask-all, content: recommendSimilarDocs(+URL, +Keywords, +Email, -UrlList)
10. Performative: ask-all, content: whoIsIntrestedInThis(+topic, +Email, -PersonList)
11. Performative: ask-all, content: find(+Keywords,’docs’, - UrlList)
12. Performative: ask , content: getRatingFor(+URL, -AverageRating, -NoOfVotes)

Any advertise messages that fall outside those outlined, are simply ignored. If the agent does not receive any of those messages, it disables any services relying on those messages until it gets the appropriate message(s).

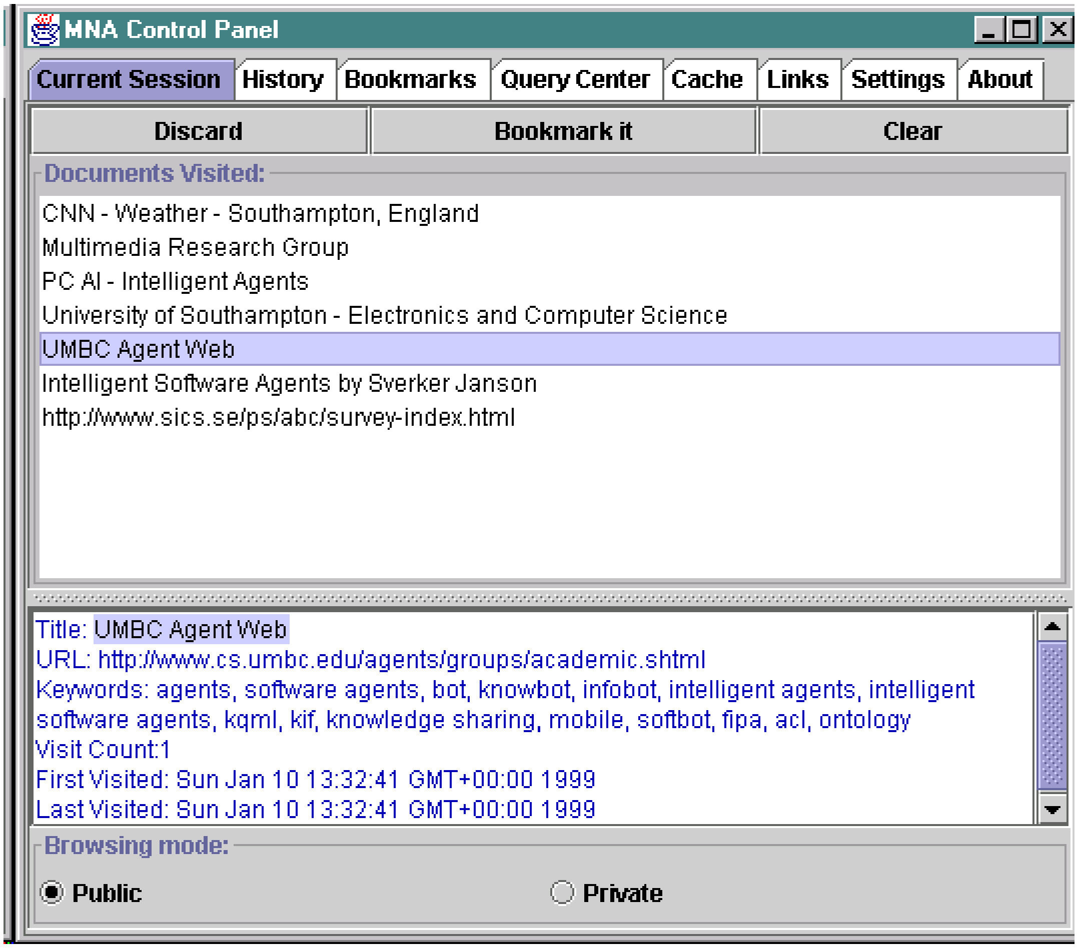

Figure 3: A snapshot of the User Interface Agent

Implementation

The user interface agent has been implemented as a Java desktop application. Figure 3 shows a snapshot of the interface offered to a user. The application is made up of many components that are essential for its operation. One of the main components of the application, is the proxy server which is used to achieve a variety of useful functions. By configuring the Web browser to use that proxy, the user interface agent can monitor the user's browsing activities and report to the organisational memory component of the system. Within the implemented system, a user can browse the Web in one of two modes: public or private. In a public browsing mode, all the user's browsing activities are reported to the OM agent. Also, for each visited URL, the proxy server creates an URL object that is used by many components of the system. The URL object, contains the title of a page, the date a page was first visited, the date it was last visited, keywords associated with a URL which are extracted using a simple extraction algorithm, and a list of image URLs contained in the page. The role of the proxy server does not end here however.

It is known that humans tend to have bad long-term memory. This is especially true when it comes to remembering URLs that they have visited and what they thought of them. To prevent users from following links to URLs that they have previously disliked and to point them to links that they have liked, the user interface agent provides the users with a facility that enables them to rate documents they are viewing. This information is stored locally as well as dispatched to the OM agent. The proxy server uses the locally stored ratings. If a rated URL is encountered as a link in any page, the proxy server adds a symbol beside that link to remind the user of what he/she thought of the page before they follow the link again.

The way in which the proxy server is built allows for incremental services that influence the display of a requested URL, to be added. A module is currently being implemented to allow agents that are capable of responding to an 'applyLinks' performative to be automatically integrated in the system. Basically, such agents would also supply a string to describe the type of links they are capable of rendering. The interface agent would then use this information to allow users to specify which links they desire rendered. For example, the image annotator agent supplies 'Links to images' type of links. In a way, the different agents that render those services play the role of filters as defined in Microcosm [5] where a message was propagated across a filter chain in which each filter could change the content of the message.

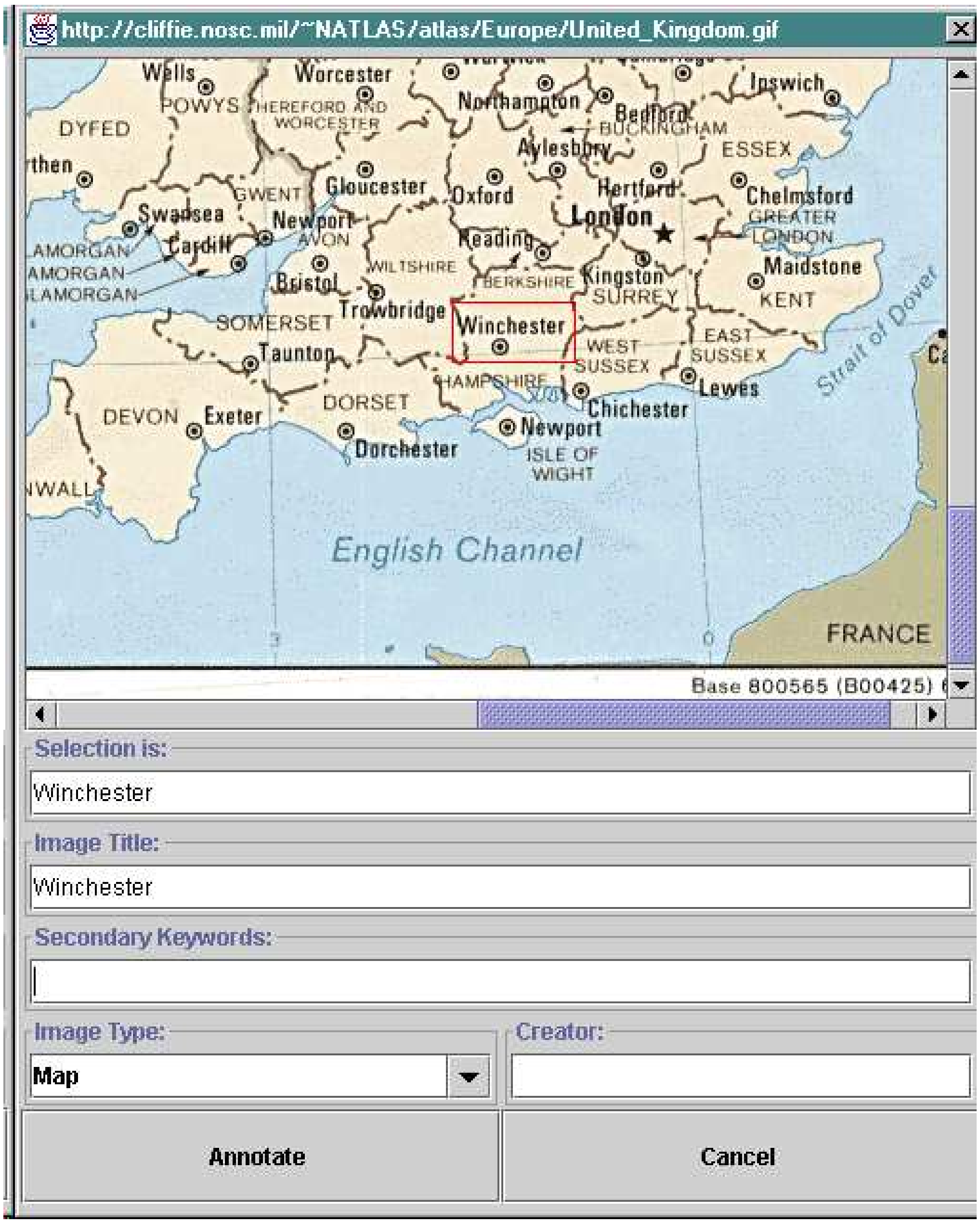

Figure 4: Snapshot of the image annotation tool

Another important component of the interface agent, is the bookmarker which allows users to bookmark documents they find interesting. Like browsing, bookmarking could be carried out either publicly or privately. This addition of privacy features enhances the usability of the system.

To increase the usability of this component, a tool to convert to and from standard bookmark files is currently being developed. The drawback to converting from standard bookmark files is that they do not provide keywords. Means, whereby keywords could be extracted automatically from imported bookmarks, are currently being investigated.

To allow users to annotate images the interface agent provides them with a tool which gives them the capability of loading and annotating image regions. The annotations entered by the users are then sent to the appropriate agent, in this case the Image annotator agent. Figure 4 shows a snapshot of the interface to that tool.

Finally, the interface agent provides an interface to all the *ask*, and *ask-all* services defined in the previous sub-sections.

4. RELATED WORK

There are many systems that have attempted to address the problem of information overload on the Web. Such systems include personal Web assistants and information filtering agents examples of which are: Webmate [4], WBI [2], Letizia [17], Alexa [12], Amalthaea [19], and InfoFinder [14], among others. Recommending a document to a user in these systems is mainly determined by the user's profile acquired by monitoring the user's browsing or keyword search activities. The presented system differs in the approach it has taken to the problem and in which bookmarks shared among users, are used to recommend documents. The presented system also differs, in the way it has integrated a number of services into a multi-agent system to facilitate information finding.

Yenta [10] is similar to the presented system in the fact that it is also a multiagent system that has been developed to match people with similar interests. The focus of that system however, is only on matchmaking and is not as open as the presented system

The system we have presented is based on the experience of the MEMOIR framework [7][21]. It is important to note that the presented system differs significantly from MEMOIR in a number of ways:

- In MEMOIR, the user interface was a non-persistent entity implemented as applets. In this system, a user interface is also a user agent which is persistent and maintains the user's context between browser sessions and offers a variety of services to the user based on his/her previous browsing experience and what he/she have expressed a liking/disliking for.
- In MEMOIR, a proxy server was used as a central component accessed by all user interface agents to monitor what users were seeing. In this system, each user agent has a proxy component which serves the same function, but which only publishes what the user identifies as public and performs other functions specific to that user. Private viewing of documents is thus achieved.
- MEMOIR employed a messaging syntax that was closely related to Microcosm's message syntax [21]. In this system, KQML messages are employed. This has contributed to greater flexibility in the agent architecture. In MEMOIR the addition of an agent required human intervention, but within this framework it can be totally dynamic.
- The goal of MEMOIR was to basically monitor the user's browsing activities and to identify their interests. In this system, this is only one of the goals. Assisting the users in browsing and locating information within other information stores, is another.

- In MEMOIR trails where identified as first class citizens. Within this system, bookmarks have replaced trails.
- In MEMOIR, an object oriented database was used as the organisational memory. In this system, a Prolog knowledge base was used for this purpose.
- Rating of documents, and making use of the rating, as well as annotating images, are services that were not available in the MEMOIR framework

The main similarity between MEMOIR and the presented system is the idea of a collaborative system. Also, bookmarks were used in a way that is quite similar to the way trails where used in MEMOIR. Collaboration in this case is achieved on two levels: the agent level and the user level. Our idea of a collaborative agent system, follows the definition of Nwana[20] where emphasis is placed on autonomy and ability to co-operate with other agents in order to fulfil designated tasks. Collaboration between users is achieved through the use of interface agents that publish aspects of the user's browsing, bookmarking, annotation, and link creation activities.

5. FUTURE WORK

There are a number of features that we would like to see implemented in future versions of our framework. These are summarised in the following points:

- The issue of security has not been appropriately addressed within the described framework. Among other things, a malicious program can impersonate an agent, register with a facilitator and start sending various agents bogus information or could ask other agents about information that is considered privileged. This of course would be a major problem if the system were to be deployed to a real world environment. Due to the experimental nature of the project, and its objectives, this was not a major concern when designing the system, but it is something that will be addressed in the future.

- In order to assist users in there information finding activities, more work on means of integrating legacy information systems that are both structured and unstructured in nature, should be carried out, and the role of ontologies in such integration needs to be further exploited

- The algorithms that were used to determine user interests, are simple and straightforward. Experimenting with more complex learning algorithms and comparing them to the ones currently applied, could provide improved results.

- We have adopted the open hypermedia philosophy through the integration with hypermedia link services. More work is planned on 'content based navigation' in this setting.

6. SUMMARY AND CONCLUSIONS

In this paper, a multiagent system that supports navigation assistance and information finding, was described. Means through which various agents inter-operate and communicate where also explained. The functionality of each of the agents and how it is represented to other agents, was detailed.

The system is currently being prepared for testing in a real life environment, and the results obtained from feedback from end users will provide the real measure to how successful the system really is. One of the organisations that will participate in the evaluation of the described system is Glaxo Wellcome, which is an industrial collaborator in this project.

From the initial evaluation experiments, several conclusions can be drawn. The described architecture is open and extensible. It supports the dynamic addition and retraction of agents and services. The fact that users can configure links to be displayed in any document based on agent advertisement of a specified message also illustrates how flexible the architecture is. Using bookmarks to recommend documents and to find out about other user interests is something that we believe is innovative and useful. It simply maximises the use of what most users do naturally. Allowing users to give feedback about what they are viewing is also a feature that seems quite useful specially when it warns users off from re-visiting pages they have already expressed a disliking for. Image annotation is yet another feature that we believe will be quite effective in increasing search hits for images.

The presented architecture simply allows a set of agents that make use of human Web navigation related activities in different ways, to share the outcome of these activities with other people in a way the assists them in navigating the Web as well as finding information.

7. ACKNOWLEDGMENTS

The work presented in this paper has been supported by ESPRC grant GR/K73060.

8. REFERENCES

[1] Altavista, 1998. http://www.altavista.com

[2] Barrett R., Maglio P. P. , and Kellem D. C. "WBI a confederation of agents that personalize the Web" , Proceedings of Autonomous Agents *'97,* pp 496 - 499, 1997.

[3] Carr L., De Roure D., Hall W., and Hill G. "The Distributed Link Service: A Tool for Publishers, Authors and Readers". Proceedings of Fourth International World Wide Web Conference: The Web Revolution, Boston, Massachusetts, USA, December 1995.

[4] Chen L. and Sycara K. "WebMate: a personal agent for browsing and searching", Proceedings of the second international conference on Autonomous agents, pages 132-139,1998.

[5] Davis H., Hall W., Heath I., Hill G., Wilkins R. "Towards an Integrated Information Environment with Open Hypermedia Systems". Proceedings of the Fourth ACM Conference on Hypertext (ECHT '92), Lucarella D., Nanard J., Nanard M. and Paolini P. (Eds.), Milan, Italy, pp. 181 - 190, December 1992.

[6] De Roure D., Hall W., Davis H., and Dale J. "Agents for Distributed Multimedia Information Management". Proceedings of the First International Conference on The Practical Applications of Intelligent Agents and Multi-Agent Technology, London, UK, 1996.

[7] De Roure D., Hall W., Reich S., Pikrakis A., Hill G., and Stairmand M. "An Open Framework For Collaborative Distributed Information Management", Proceedings of Seventh International World Wide Web Conference (WWW7), Brisbane, Australia, April 1998.

[8] El-Beltagy S., Hall W. and De Roure D. "Tools for Facilitating Linking to and Searching for Image Regions on The Web", in Computer Science Technical Report, University of Southampton, UK, ECSTR M99/1.

[9] Finin T., Fritzson R., Mckay, D., and McEntire, R. "KQML- A Language and Protocol for Knowledge and Information Exchange." University of Maryland, UMBC, Technical Report CS-94-02, 1994.

[10] Foner L. N. “A Multi-Agent Referral System for Matchmaking”. Proceedings of the First International Conference on The Practical Applications of Intelligent Agents and Multi-Agent Technology, London, UK, 1996.

[11] Gudivada V. N., Raghavan V. V., Grosky W. I., and Kasanagottu R. “Information Retrieval on the World Wide Web”, *IEEE Internet Computing*, 1(5): 58-68,1997.

[12] Kahle B. and Gilliat B. “Alexa - navigate the Web smarter, faster, easier”. Technical report, Alexa Internet, Presidio of San Francisco, California, USA, 1998.

[13] Khedro T, and Genesereth M. R. “Facilitators: A Networked Computing Infrastructure for Distributed Software Interoperation” Presented at the Workshop on AI in Distributed Information Networks, IJCAI-95, Montreal, Quebec, 1995. Available from: http://www.stanford.edu/group/CIFE/FCDA/

[14] Krulwish B., and Burkey C. “The InfoFinder Agent: Learning User Interests through Heuristic Phrase Extraction”. IEEE Expert/Intelligent Systems & Their Applications 12(5): 22-27, 1997.

[15] Labrou Y. and Finin T. “A Proposal for a new KQML Specification.” University of Maryland, UMBC, Technical Report CS-97-03, 1997.

[16] Lesser V. “Multiagent Systems: An Emerging Subdiscipline of AI”. ACM Computing Surveys, 27(3): 340-342, September 1995.

[17] Lieberman H. “Letizia: An Agent That Assists Web Browsing”, Proceedings of the 1995 International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995.

[18] Moreau L., and Gray N. “A Community of Agents Maintaining Link Integrity in the World Wide Web”. PAAM98 - The Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents., Nwana H. S. and. Ndumu D. T (Eds.), London, UK, 1998.

[19] Moukas A., Zacharia G. “Evolving a Multi-agent Information Filtering Solution in Amalthea” Proceedings of Agent’ 97, Marina Del Rey, California, USA, 1997.

[20] Nwana H. S. “Software agents: an overview”, The knowledge Engineering Review, 11(3), pp. 205-244, 1996.

[21] Pikrakis A., Bitsikas T., Sfakianakis S., Hatzopoulos M., DeRoure D, Hall W., Reich G, Hill G, and Stairmand M. “MEMOIR - Software Agents for Finding Similar Users by Trails”, PAAM98 - The Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents. Nwana H. S. and. Ndumu D. T (Eds.), London, UK, 1998.