Agent Based Internet (WWW) Services

László Gulyás, László Kovács,
András Micsik, László Tersztenyák

MTA SZTAKI
Computer and Automation Research Institute
of the Hungarian Academy of Sciences
Department of Distributed Systems
H-1111 Budapest XI. Lágymányosi u. 11. Hungary
gulya@dennis.inf.elte.hu
{laszlo.kovacs, micsik, tersztenyak}@sztaki.hu

Abstract

The evolution of the World Wide Web heads towards the use of more and more intelligent services. A possible solution to implement this service oriented view of World Wide Web is to use agents. A framework is proposed to provide agent-based Internet services via WWW. A case study introduces the actual use of this framework and its relation to the agent technology. As an implementation effort the Personalized Home Page (PHP) system is presented here.

Keywords: autonomous agent, multi-agent system, internet agent, World Wide Web, customization, service

1 Introduction

The current usage of the Internet is based on accessing and downloading pieces of information. Documents or data may be generated on-line, but that is hidden inside Internet servers. From the user's point of view this is a quite simple cooperation pattern. All they can do is to search and bookmark locations on the Web, and select appropriate locations to access the required information. Some WWW services provide customization methods to their users. Presentation modes or information preferences can be chosen. Thinking further in this direction a new service-oriented WWW environment can be imagined, where intelligent services are offered to the user instead of static documents.

Agent technology [5] offers the best background for the implementation of service-oriented WWW. The Internet is an extremely promising area for the deployment of agent technology. It can provide a rich environment with lots of information distributed over large number of hosts. On the other hand the Internet needs more powerful technologies to cope with its information load.

The structure of the paper is the following: the concept of service-oriented WWW is examined in Section 2. A framework for service-oriented WWW is introduced in Section 3, and the implementation of Personalized Home Page (PHP) system based on this framework is detailed in Section 4.

2 Service oriented World Wide Web environment

Presently the information available on the Internet is used rather passively. In the everyday cyberspace life Web users use many Web services at the same time. They can find these services using home made or public bookmark (URL) collections or using search engines. After finding the URLs of required services their pages will be loaded in. Users usually use only parts of services. Therefore the unused parts of service pages are unnecessary loaded in. The parallel use of different WWW services makes users to exchange different pages within the same window or use some overlapping windows. Screen and browser resources are used uneconomically.

There should be more ways for a user to affect the operation of Internet information services. In particular users should be able to:

- select and construct their own views of internet services
- customize internet services for their own preferences
- execute interactive and automated tasks over the Internet

A service can be a collection of static HTML pages, a Java applet, or dynamic pages generated by a program, or a mixture of these. In our view there is a wide range of functionality that internet services can provide. A simple service just shows an HTML document, a more complex one provides an online dictionary or translation service. Some services cooperate with each other to accomplish tasks involving several hosts or networks. Examples of cooperating services are to search information about something, to find a color printer on the local network, or to get a differential equation solved.

In this view hosts offer sets of services. Users have the possibility to select, configure and use these services on the network. There are two important questions in this scenario. First, how will the services cooperate with each other? Secondly, how will they communicate with the user? Cooperation issues involve the identification of services, finding other services on the network and exchanging information between services.

Tasks given to services can be executed immediately or in the background. Background operations can be continuous or timed. Services can interact with the user during execution, they can ask questions, give warning messages, and list results. Therefore a service must be able to present things to the user. On the other hand the user should be able to arrange the presentation areas of services as he likes.

To fulfil the above requirements, an agent-based [1, 2] software solution is suggested in this paper. Agent technology can be used to implement internet services, and one or several agents can be seen as a service. In our language an agent is a piece of software which can communicate with other agents. It is able to detect and discover the local environment (e.g. local agents, system resources) and other agents on the Internet. Furthermore it has a certain (usually user-given) goal to achieve and it really acts in a way leading towards that goal. These actions can be reactions to events in the environment or can be originated from the internal state of the agent.

3 Framework

The generic architecture of this agent framework is presented in Figure 1. The framework consists of a set of user, service and network agents. There are specialized user and service agents (agent personalizer, mediator agents). Their role is to manage other agents. Detailed descriptions of agent types are given below.

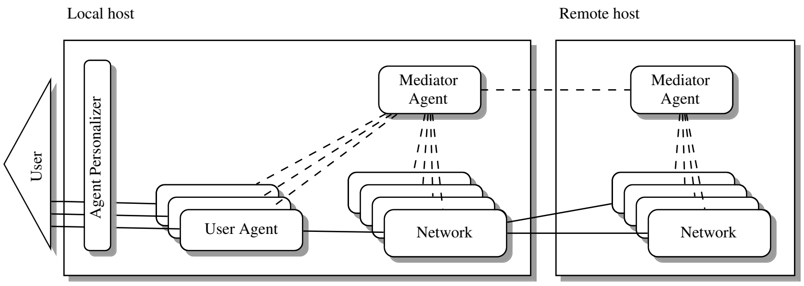

Figure 1: Overview of the agent framework

The basic operation of this framework:

- A user configures his user interface for the User Agents through the Agent Personalizer.
- The chosen User Agents then can be started, and the results of their work are displayed for the user.
- User Agents find appropriate Network Agents to cooperate with.
- This finding and the communication between all agents is helped by a special Mediator Agent.

3.1 Basic Entities

3.1.1 User Agents (UA)

Each User Agent must have a way to communicate with the user. This means that it drives a user interface complying to the possibilities offered by the Agent Personalizer, and provides methods to configure its operation through the Agent Personalizer.

User Agents can communicate with other agents in the framework. If a User Agent wants to participate in a conversation over the Internet, it may rely on a local Network Agent to do the conversation on behalf of it. During its operation the entire User Agent can move to the host where the user interface is displayed.

3.1.2 Agent Personalizer (AP)

The main role of the Agent Personalizer is to help the users to display agents on their screen. Each User Agent has a dedicated piece of the display where it can talk to the user, and this piece is directly controlled by the User Agent. However there are several other tasks remaining for the Agent Personalizer:

- to arrange the display pieces of the User Agents on the screen
- to store the personal configuration of the individual users
- to keep a database of the available User Agents and their properties
- to provide possibility for the user to create, delete or modify display scenarios, and to add, delete or configure User Agents inside the scenarios.

The user chooses from the UAs, and configures each by setting the display and functionality options. The chosen UAs are arranged on the screen of the user, and this scenario is fixed inside the Agent Personalizer, so it can be loaded on the screen any time the user wishes.

The Agent Personalizer can be regarded as an agent itself, so it can participate in conversations with other agents. For example a User Agent can register itself at the Agent Personalizer. As part of the registration it has to describe itself, its configurability, its demands for the display area.

3.1.3 Network Agents (NA)

Network Agents typically do not have user interfaces. These agents communicate mostly with each other, or with User Agents.

Service Agents are specialized Network Agents: they provide some services for other agents, and therefore they are accessible from remote agents. They can provide some kind of information, or undertake different tasks such as sending a letter or calculating an expression. For Service Agents the same characteristics are desired as for any LAN or Internet services: continuous availability, parallel servicing, good data throughput, etc.

Network Agents which do not fall into the service category are performing their own tasks without any regulations for availability and life time. Some of those can be operating only while the task is not accomplished.

Network Agents know how to communicate with other agents over the Internet with the help of Mediator Agents. They register at their local Mediator Agent giving the information about their capabilities and access modes. However the details of communication techniques may be transparent for these agents, since their needs are served by Mediator Agents. This simplifies the construction of such agents.

3.1.4 Mediator Agent (MA)

Mediator Agents provide the most important facilities for other agents: trading of agent services. The basic idea about their operation is that there must be a per host coordination point for agent communication on the Internet. These Mediator Agents help Network Agents to find each other and to build a channel for communication. The communication itself need not necessarily flow through the Mediator Agents.

The process when two Network Agents begin to communicate with each other is detailed now.

1. The agent who initiates the conversation gives a required capability list to the local Mediator Agent to find a remote agent satisfying the list.
2. The local Mediator Agent finds some agents meeting the requirements during a compound query process in collaboration with other Mediator Agents. Mediator Agents use learned and programmed considerations about solving this problem most economically.
3. The initiating agent chooses one from the list of found agents (this will be called the invoked agent).
4. The Mediator Agents on the host of the initiating and invoked agents communicate with each other to agree in the following issues:
    - whether the communication of the two Network Agents is allowed or not,
    - what should be the properties of the communication channel between the two Network Agents.
5. Both the initiating and invoked agents are notified about the communication possibility, and the communication channel is built.

The main tasks of a Mediator Agent include:

- to keep a database of available Network Agents,
- to cooperate with other Mediator Agents in searching desired agents,
- to help in building and maintaining communication channels between agents.

The first point indicates that there is a need to have a description language for agents, where as the most important part of this language agent capabilities can be given. The most often case is where agents search for other agents according capabilities. The second indication is the need for naming and identifying agents.

3.2 About the agents in this framework

By the term of agent usually a neverending process is mentioned (see Stan Franklin's and Art Graesser's definition in [3]). Nevertheless a piece of software acting for a considerable amount of time can produce many features of agency. Agents in this framework can be launched on demand by the Mediator Agent, which creates great flexibility in agent scheduling.

All agents in this framework take on some kind of responsibility from others or from the user. Usually User Agents take theirs from the user while Network Agents and Mediator Agents from the others. To achieve their goals and fulfill their duties the agents do not necessarily need high level of intelligence but they need the power of communication. They can communicate with three possible partners: users, local agents and remote agents. Typically User Agents are involved in dialogues with users, and Network Agents communicate with remote agents, but there may be agents that fall into both categories. The communication can be asynchronous as well. For example an agent may not be reachable in a certain part of the day, but it may receive tasks through e-mail.

Some agents in order to perform their tasks move between computers. Specially this is the case with User Agents which appear on the user's remote display. Though User Agents are the usual source of mobility in this framework, the Network Agents can also move with the Mediator Agents serving as the entry point for a visit. However this mechanism is not detailed in this paper.

3.3 Communication of agents

This framework is aiming to provide a general approach so it cannot discuss the deep details of the Agent Communication Protocol. The communication channel between agents is constructed by the Mediator Agents. The connected agents can exchange messages. These messages maybe written in any Agent Communication Language, the understanding of each other is the problem of the talking agents. There are some standardized messages providing basic possibilities for agent cooperation.

Agents has names, addresses and capabilities. Addresses are used to construct the communication channel. Names uniquely identify agents in the world. Capabilities describe the services and roles of the agents. The capability scheme sets how the agent can declare its abilities, access restrictions, and other properties.

Agents can search for other agents with desired capabilities. This is supported by the Mediator Agents and capability queries. Mediator Agents are expected to cooperate with other Agents to give a starting help for searching agents. The exact details of this agent exploration is not fixed, rather it is promoted by a standardized query mechanism.

3.4 Relations to other efforts

3.4.1 Searching

Nowadays the most developed search methods over the Internet are agent-based. Among these we can find personalized agents learning their user's interests and proposing possible URLs or multi-agent systems answering the user's queries by extensive communication based on their knowledge and reasoning.

Waldo the Web Wizard developed by Andersen Consulting [8] is an example for the first case. This agent acts upon the knowledge about the user's interests explored by an other agent (LifestyleFinder).

The Do-I-Care agent [9] is an innovative World Wide Web agent that uses a model of collaboration to leverage the natural incentives for individual users to easily provide for collaborative work. Another information gathering agent is Ingram (developed in the Softbot project [10], driven by the Occam forward chaining planner.

3.4.2 Agent frameworks

The Open Agent Architecture (OAA) is a cooperative framework for agents [6]. It can be used to construct distributed systems. The communication of agents is based on the Interagent Communication Language which is specific to OAA, and similar to KQML [4]. OAA has a common data repository, and a delegation service for agents based on goals defined in ICL. The most interesting part of OAA is its capability to handle multimodal user interfaces. However OAA feels as a closed environment, and is not specifically prepared for WWW integration.

IBM has implemented a development kit for aglets [7], which are agents implemented in Java. Aglets are essentially mobile agents, they can move to hosts that provide the Aglets Framework for them. The Aglets Framework covers agent naming and collaboration as well. Interfacing of aglets with the user is used by individual applets, but there isn't any extensive graphical user interface for aglets.

3.5 Examples of Agents

Here we show by examples how solutions for everyday computing needs can be solved using agent based Internet technology.

3.5.1 Simple User Agents

As examples for simple User Agents a clock or a desktop calculator can be mentioned. The clock starts by displaying a clock-face and then periodically updates the clock-face to show the current time. This is a simplest kind of a User Agent, it has a one-way communication with the user (showing the time), and some simple configuration options (clock-face selection, etc.). As a User Agent it supports the protocol towards the Agent Personalizer, thus the Agent Personalizer can offer the Clock Agent to users, and can perform the configuration if the user wishes. At activation the Clock Agent gets its configuration, moves to the displaying host, and keeps running there. This agent can be enhanced with inter-agent communication to show the time of other hosts (time servers). In this case Network Agents would be deployed to periodically acquire the time from the selected host. With further enhancement the clock could itself find an appropriate Service Agent to get the right time from.

3.5.2 Simple Network Agents

In this sense regular Internet servers (WWW, Gopher, Finger, etc.) are primitive Service Agents, because while they are not aware of communication possibilities with other agents, they show up a lot of basic agent properties. Thus any regular TCP/IP services can be integrated to our system as Service Agents. This is done by registering the services as Network Agents and specifying their capabilities. Capabilities and properties give enhanced search and selection possibilities to the user to find the best service according to his needs.

3.5.3 Communicating Network and User Agents

Watchdog The Watchdog agent can be configured to report the presence of a user on a set of hosts. It periodically tests if the user has logged in on any of the given hosts.

The Watchdog User Agent has to find Service Agents which can inform it about the currently working persons on their hosts. The agent is started on the displaying host with the help of the Agent Personalizer. The agent then asks the Mediator Agent on its originating host to get information from the Finger Agents on the watched hosts. The agent could also turn to the Mediator Agent on the displaying host if it was allowed, but that solution is not secure regarding the displaying host. In this case there is no need for capabilities, the agent knows the name of the desired Service Agent. Finger Agents provide the functionality of a finger daemon. The Watchdog Agent then periodically queries finger information, filters it and displays the results. The Finger Agent however can be improved to deny to service given agents, or to give different information depending on the agent that asks (a kind of good secretary).

Mirroring Mirroring in the Internet jargon means to replicate information or services and make those available on other hosts. The process of mirroring uses data access protocols (FTP, WWW) to gather data, and then reconstructs the remote information service locally from the downloaded information pieces. The mirroring of complex information services on the Internet can be a rather difficult task, and in current usage the mirroring of WWW services is automated by individual scripts, each one is applicable for the mirroring of a certain WWW service.

The problem is that part of the data needed in the copy of a WWW service is reachable only locally from the host holding the original WWW service. These tasks could be easily translated into agent systems, where Mirror Service Agents provide mirroring facilities for Mirroring Agents. Mirror Service Agents can access data locally and forward to Mirroring Agents residing on the host where the replica is to be built. In this way all help can be given to Mirroring Agents to create the replica, and it can be controlled as well.

Details of possible mirroring processes are found in [13], where similar approach is discussed for the Dienst Digital Library System. Here the construction of WWW mirroring agents is discussed expanding the mirroring environment developed at SZTAKI [12].

A scenario of mirroring is started by the user when he gives the root URL of the document to be mirrored, and sets his mirroring preferences. Then the Mirroring Agent contacts the Mediator Agent at the host from where the mirror has to be done through his own Mediator Agent, and asks for an agent capable to mirror the given document. There can be several Mirror Service Agents on a host, each providing mirroring facility to different information services. If the right Mirror Service Agent is found, it decides whether to allow the creation of the copy, and then creates a portable format of the document and sends it to the Mirroring Agent. The Mirroring Agent creates the copy locally and notifies the user. During this both the Mirroring Agent and the Mirror Service Agent may decompose his task and delegate them to several agents.

4 The Personalized Home Page System

The Personalized Home Page system (PHP) is a running prototype of the agent-based World Wide Web services framework developed at MTA SZTAKI, the Computer and Automation Research Institute of the Hungarian Academy of Sciences [11].

In PHP the WWW services are provided by User Agents. In the current implementation any Java-capable WWW browser is appropriate as the display area of

User Agents. This choice adds an extra enhancement to the network operation of agents, since the display need not be on the same machine where the User and Agent Personalizer agents are. Two kinds of display format are allowed: HTML or Java applet.

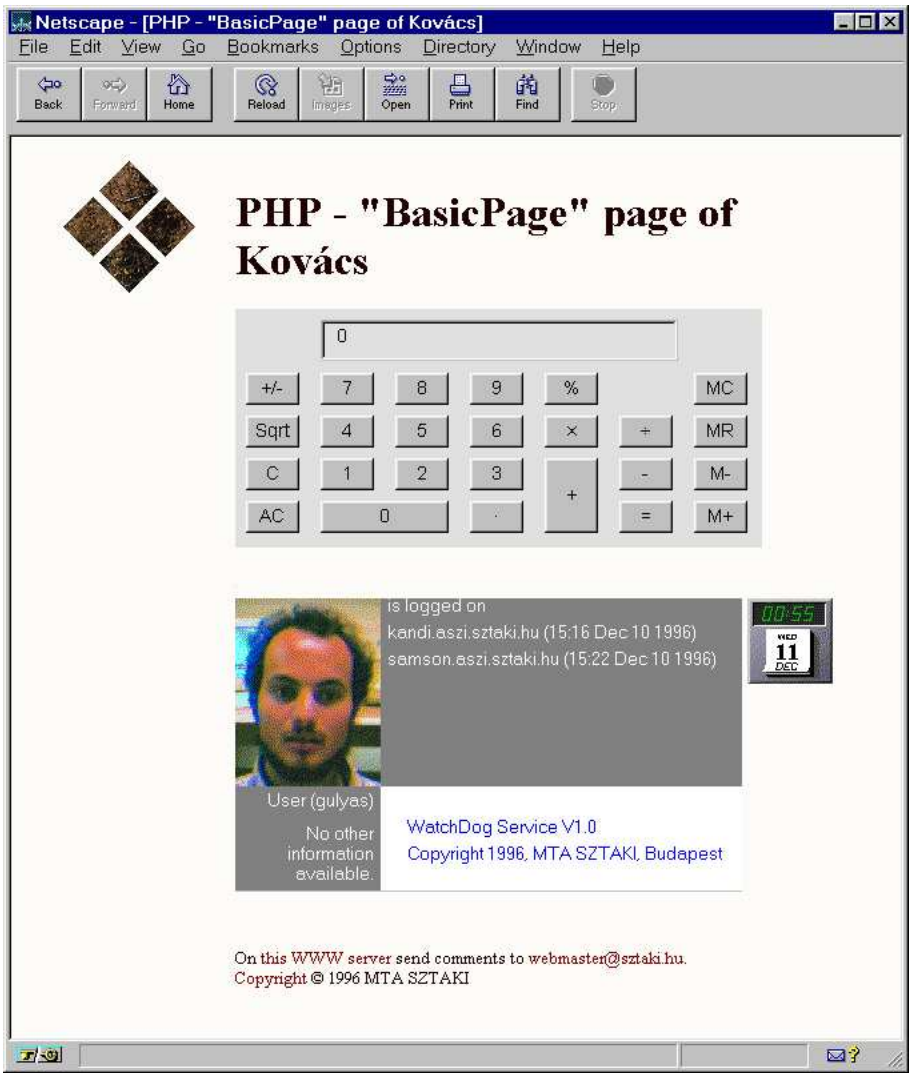

Figure 2: PHP page constructed by a user

4.1 Personalized Virtual URLs

In PHP individual users can select the required WWW services and can build their own (virtual) pages from these selected services. This can be done on-the-fly. In the running prototype a set of virtual pages can be defined. The URLs of these user-defined pages can be given by the user himself. Their uniqueness for a particular user is guaranteed by the PHP system. In the current PHP implementation the last part, the path part can be defined by the users as the first part of the virtual URLs is PHP predefined. E.g. in the case of *http://www.sztaki.hu/php/MyServices/MySearchPage* virtual URL the *http://www.sztaki.hu/php/* is the predefined PHP system prefix and the MyServices/MySearchPage is the user-given path. A user can have as many own virtual pages as he wants.

After the definition of the virtual URLs the user can construct the content of these pages selecting the services, the User Agents from the available agent pool. The layout of the virtual pages, e.g. the position and size of the windowing areas of User Agent applets can be influenced by the users as well. The user can add or delete agents in a page, and change the configuration of the agents. Agent configuration includes display options and parameters passed to the agent. The PHP system behaves as an Agent Personalizer in this respect.

User Agents are started when the user downloads one of his personal pages to a WWW client. PHP provides the previously set UA parameters. Some UAs present a form-based interface, and they are active only for the period of time to answer the request specified by filling the form. Others may be active continuously (e.g. Java applets).

In PHP security can be set up in two ways: the page can be protected by password, or its access can be limited to one host (IP address). The latter case is preferred in our intranet environment where workstations for individuals can automatically download a personalized home page of the particular user without any further intervention.The current PHP system is implemented in Perl, and agents are implemented in Perl and Java.

4.2 Creation and usage of a sample PHP page

During the creation process the user communicates with the user-, and page-maintenance parts of the PHP system in order to maintain his personalized pages. Users can manage their pages in the PHP system from the PHP starting page. There are two choices: to create new virtual pages, or to maintain the existing ones.

For a new page, the title and the virtual URL of the page must be given. URLs of user constructed pages start with the standard prefix within PHP system. New users must also choose a password. Access configuration of page "BasicPage" has two options: to allow access to the page from a given host, or to allow access after giving the user's password. The first option is to ease the work with PHP

Figure 3: Changing the view of services

pages from computers in personal use. After these steps the user can construct the page, using the same methods as for maintenance.

Page maintenance starts with user authentication, and the pages of the authenticated user are offered for modification with the following possibilities: change access permissions, modify the content, or delete the page.

Under content modification (Fig. 3) user can change, delete or add new services (User Agents). If the service has parameters, they are changeable here as well. Parameters of User Agents are managed by HTML forms generated by the system. The process of configuration is controlled totally by the service itself. The parameters resulting from the configuration process is handed over to the PHP system, which stores it.

Adding a new service means the selection of the service to add from the available service (User Agent) pool. There are additional HTML tags to improve the layout of the user interface areas of selected User Agents.

Figure 2 shows the personalized page called "BasicPage" of user Kovacs with the selected WWW services: calculator, watchdog, clock. This page can be downloaded from the PHP Web server e.g. using the *http://www.sztaki.hu/php/BasicPage* virtual URL and the correct password.

5 Summary

A new service oriented view of WWW was given. WWW services are provided by active entities, agents. The architecture of agent-based WWW was discussed including the features of different agent classes and the software glue to build the framework.

The proposed agent framework was successfully used in the building of PHP, the Personalized Home Page system of MTA SZTAKI. PHP opens a new direction to the application of individually customized dynamic Web pages.

References

[1] M. Wooldridge and N. R. Jennings, "Agent Theories, Architectures, and Languages: A Survey" in Intelligent Agents ECAI-94 Workshop Proceedings; Lecture Notes in Artifical Intelligence 890, Springer-Verlag, Berlin, 1995.

[2] Hyacinth S. Nwana , "Software Agents: An Overview", Knowledge Engineering Review, Vol. 11, No 3, pp.1-40, Sept 1996. Cambridge University Press, 1996

[3] S. Franklin, A. Graesser: Is it an Agent or just a Program?: A Taxonomy for Autonomous Agents, Working Notes of the 3rd International Workshop on Agent Theories, Architecture and Languages held at 12th ECAI, Budapest, Hungary, August 12-13, 1996

[4] J. Mayfield, Y. Labrou, and T. Finin: Evaluation of KQML as an Agent Communication Language, Intelligent Agents Volume II – Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages Lecture Notes in Artificial Intelligence, Springer-Verlag, 1996. 11/7/95

[5] "Intelligent Agents White Paper" by Don Gilbert, Manager, IBM Intelligent Agent Center of Competency, with Pete Janca, URL: http://www.raleigh.ibm.com/iag/iagwp1.html

[6] D. L. Martin, A. J. Cheyer, and D. B. Moran, "Building distributed software systems with the open agent architecture," in Proc. of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, (Blackpool, Lancashire, UK), March 1998

[7] IBM Aglets Software Development Kit, URL: http://www.trl.ibm.co.jp/aglets

[8] Waldo the Web Wizard (developed by Andersen Consulting), URL: http://bf.cstar.ac.com/lifestyle/lf_dis.htm

[9] Brian Starr, Mark S. Ackerman, Michael Pazzani: "The Do-I-Care Agent: Effective Social Discovery and Filtering on the Web", Proceedings of RIAO'97 (Computer-Assisted Information Searching on the Internet), pp. 17-31.,
URL: http://www.ics.uci.edu/ãckerman/docs/97.riao/dica.riao.html

[10] The Ingram softbot,
URL: http://www.cs.washington.edu/homes/ctkwok/ingram.html

[11] L. Gulyás, L. Kovács, A. Micsik, L. Tersztenyák: Personalized Home Pages - WWW Services Based on Agent Technology, SZTAKI Technical Report TR97-3, February 97

[12] L. Kovács, A. Micsik, G. Schermann: An Environment for Mirroring Hypermedia Documents, Joint European Networking Conference (JENC) 7, Budapest, May 13-16 1996.

[13] L. Kovács, A. Micsik: Replication within Distributed Digital Document Libraries. Proceedings of the 8th ERCIM Database Research Group Workshop on Database Issues and Infrastructure in Cooperative Information Systems, Trondheim, Norway, 1995

[14] L. Kovács: The GroupSPACE Concept, Proc. of the 14th IEEE International Conference on Distributed Computing Systems (ICDCS-14), Poznan, Poland, 1994