# PERSONALIZED HOME PAGES - A WORKING ENVIRONMENT ON THE WORLD WIDE WEB

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

## Abstract:

*The World Wide Web is one of the most common interfaces to the Internet and thus to the global office as it provides an easy-to-use and self-explaining user interface for teleworkers. However WWW based interfaces are relatively rigid and are lacking ways of user customization or setting preferences. The new service oriented approach presented here builds upon the strength of the WWW interface to Internet services but enhances it by the power of individual customization. The Personalized Home Page (PHP) system is built on a framework capable to describe all Internet services in a uniform way using the terminology of weak agency. This paper gives an overview of the PHP system with a short description of the agent based framework behind. Examples introduce the actual use of the system.*

## Keywords:

Teleworking environment, WWW service, customization, dynamic Web page, PHP, multi-agent system

# 1. Introduction

The World Wide Web is one of the most common interfaces to the Internet and thus to the global office. This technology offers a good base for any application since users face with a standard, easy-to-use interface. As teleworkers must be provided with at least as user friendly tools as they can find in traditional offices, the complexities of the global network should be hidden. Although WWW technology makes a very significant step in this direction, it still requires a good set of technical knowledge to exploit the possibilities of the global network. It is still required to know addresses, ports, URLs and other technical details which can be a hard job to learn from time to time.

The current usage of the Internet is based on accessing and downloading pieces of information and applications. This usually means to surf over the network several times a day, often in several browser windows and using mostly only a part of the information provided on the accessed webpages. This process is unable to hide the inherited complexity of the global office. Well-designed and highly focused WWW sites may help a lot but they also take away the freedom of using the entire network as office resources.

The Personalized Home Page (PHP) system developed at MTA SZTAKI addresses these issues by offering an individually customizable interface to Internet and WWW services as presented in section 2. The motivation and several detailed examples are also discussed there. The key to the offered solutions is an agent based description of Internet services which builds up a general framework as summarized in section 3.

# 2. PHP - Personalized Home Page System at MTA SZTAKI

The Personalized Home Page System (PHP) is a running prototype of a new teleworking environment recently developed at MTA SZTAKI, the Computer and Automation Research Institute of the Hungarian Academy of Sciences. This new environment interfaces Internet services to the World Wide Web based on an agent based general framework.

## 2.1 Motivation

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 downloaded. Users usually use only parts of services/pages. 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. Moreover the user is never allowed to influence the content and the layout of HTML pages or the output of network applications she uses.

Let's take the example of a user who wants to access two Web-based applications: a calculator located at site A, and a simple utility at site B that monitors the local network and reports the presence of a user, in this case one of her colleagues. Both services are available on the network, and are accessible through standard browsers. However they are located at different sites thus the user is required to open up two WWW clients, remembering or at least selecting the appropriate URLs from her bookmark collection. Moreover she does not have the opportunity to customize the location, color, initial parameters and other properties of these two applications.

Trying to overcome these difficulties the work reported in this paper is based on a concept that sees WWW servers as collections of services rather than collections of HTML pages. According to this view users do not download individual pages anymore but access a set of previously customized WWW services, that is a complex unit of service, one of her Personalized Home Pages.

**2.2 Features of PHP**

According to the concept of PHP WWW servers provide services and not individual HTML pages anymore. A service can be a collection of static HTML pages, a Java applet, dynamic pages generated by a program, or a mixture of all these. Users do not download individual pages from a WWW server anymore but use available complex WWW services. This service concept is nothing more but a compact packing of current WWW technology.

In PHP individual 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 Agents reside. Two kinds of User Agent output display format are allowed: HTML or Java applet.

2.2.1 Personalized virtual URLs/pages

In PHP individual users can select the required WWW services and can build their own (virtual) pages from these selected services. The URLs of these user-defined pages can be given by the user herself. 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, while the first part of the

virtual URLs is PHP predefined. E.g., in the case of the *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 she wants.

### 2.2.2 Service selection

After the definition of the virtual URLs the user can construct the content of these pages selecting the services, that is 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 user 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. Configuration of agents is supported by another agent called Agent Personalizer. This agent behaves as a User Agent but the main task of it is the configuration of other User Agents.

### 2.2.3 Service use

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

In PHP privacy can be ensured in two ways: a virtual 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.

Running User Agents provide their services as a result of a cooperation with other agents, called Network Agents. Network Agents can be found with the service of a specialized agent called Mediator Agent. Detailed description of the operation of PHP is given in section 3.

## 2.3 Creation and usage of a sample PHP page - Examples

In the following a complete process of creation and use of a Personalized Home Page is presented. During the creation process the user communicates with Agent Personalizer in order to maintain her personalized pages.

Users can manage their virtual pages from the PHP starting page (Figure 1). There are two choices: to create new virtual pages, or to maintain existing ones. For a new page, the title and the virtual

URL of the page must be given. (Figure 2). After these steps the user can construct the page, using the same methods as for page maintenance.

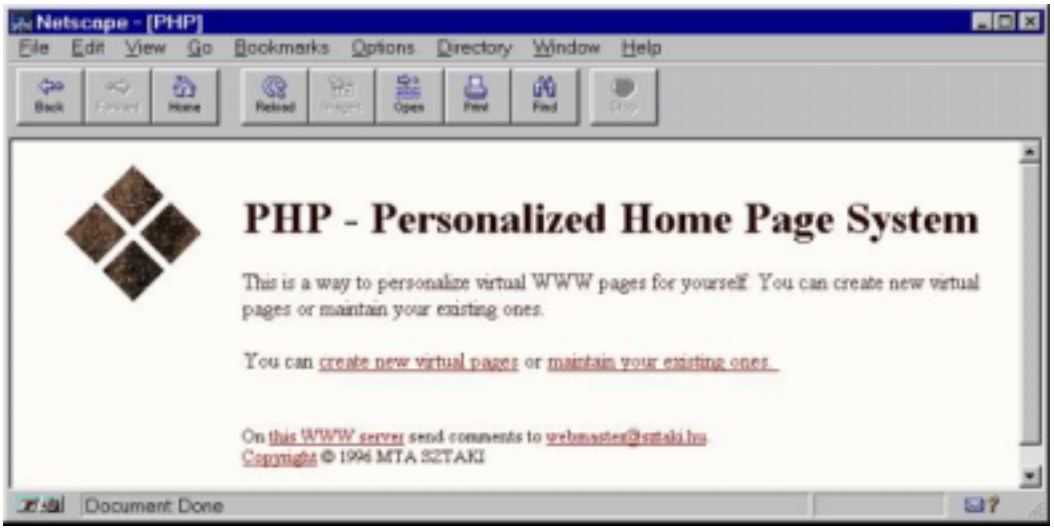


**Figure 1**

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 user can change, delete or add new User Agents (services). If a service has parameters, they can be changed 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 are 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. (Figure 3) In the pool there are locally available user agents and additional HTML tags to improve the layout of the page. If a User Agent is not available locally first it has to be found using a Search User Agent. After finding the required User Agent somewhere in the network it can be used locally with its registration. In the current PHP implementation this User Agent registration is a semi-manual process but in the future an intelligent Network Agent can be written for this purpose as well.

Figure 4 shows the personalized page called “BasicPage” of user Kovács with the selected WWW services: calculator, watchdog, and clock. This page can be downloaded to host *ovid* from the PHP system improved WWW server e.g., using the *http://www.sztaki.hu/php/BasicPage* virtual URL.

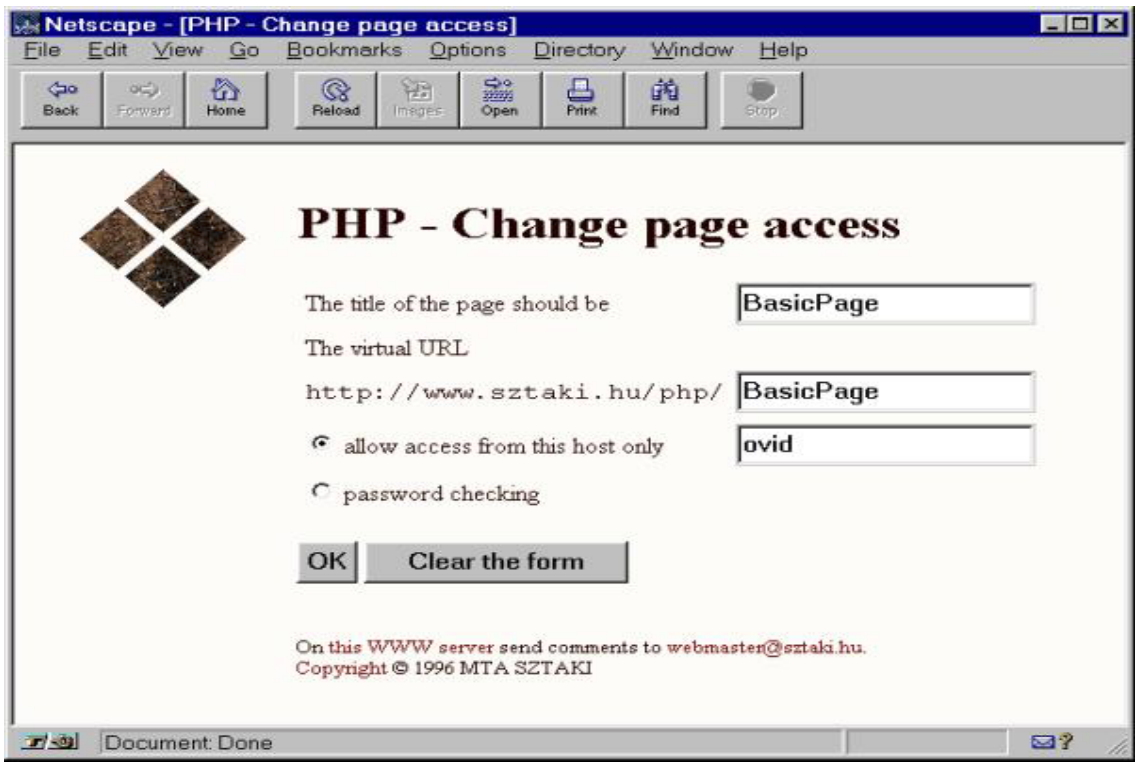


**Figure 2**

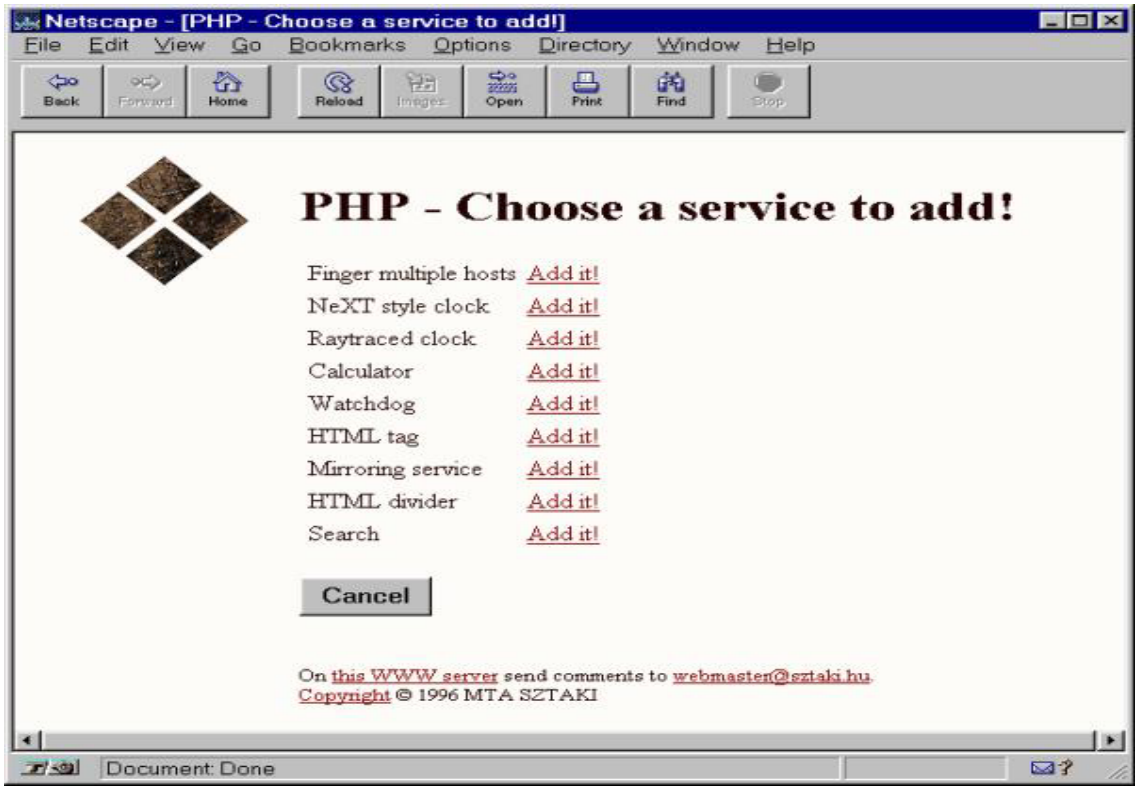


**Figure 3**

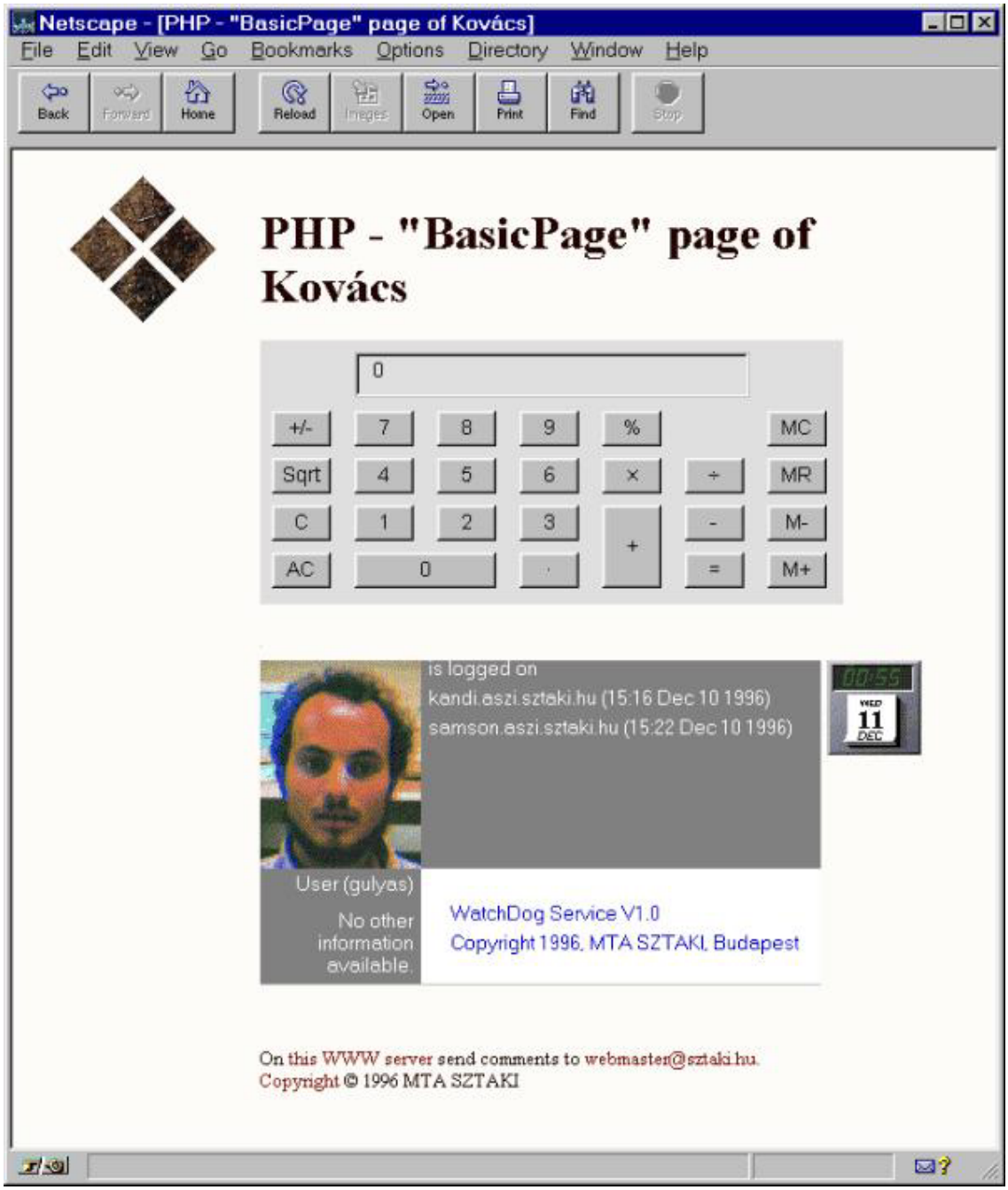
Netscape - [PHP - "BasicPage" page of Kovács]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
PHP - "BasicPage" page of Kovács
0
+/- 7 8 9 % MC
Sqrt 4 5 6 × ÷ MR
C 1 2 3 + - M-
AC 0 . = M+
is logged on
kandi.aszi.sztaki.hu (15:16 Dec 10 1996)
samson.aszi.sztaki.hu (15:22 Dec 10 1996)
User (gulyas)
No other information available.
WatchDog Service V1.0
Copyright 1996, MTA SZTAKI, Budapest
On this WWW server send comments to webmaster@sztaki.hu.
Copyright © 1996 MTA SZTAKI

**Figure 4**

## 3. The Underlying Framework

The PHP system described in the previous section operates over a general, agent based [4] framework which makes the system expendable, flexible, and appropriate to include every existing WWW services. The framework consists of a set of interface and collaborative agents in the sense of [7]. These are the user, service and network agents. There are specialized user and service agents (Agent Personalizer, Mediator Agent). Their role is to manage other agents. Figure 5 shows an overview of the framework.

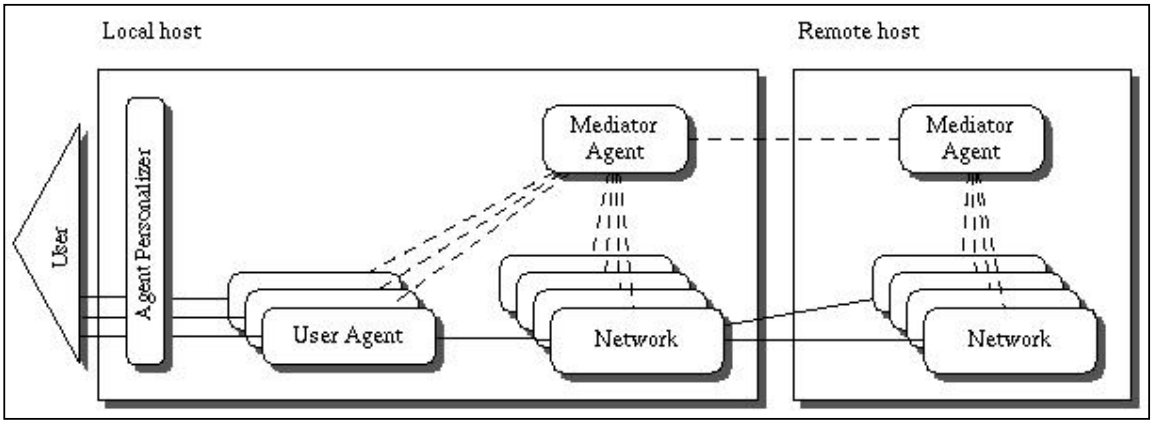


**Figure 5**

The basic operation is as follows: the User Agents on the user's PHP page start and display their results in the browser according to the configurations previously set using the Agent Personalizer. During their operation User Agents find appropriate Network Agents to cooperate with. The cooperation and communication is facilitated by special Mediator Agents.

### 3.1 Basic Entities

The basic units of the system are agents in the term of *weak agency* [8] as they carry out tasks autonomously on behalf of the user or other agents [3]. The system's agent classes are described below.

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 may communicate with other agents over the Internet relying on a local Network Agent. 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 window where it can talk to the user, and this piece is directly controlled by the User Agent. However there are several other tasks to be performed by the Agent Personalizer such as to arrange the display pieces of the User Agents on the screen, to store and update the personal configuration of the individual users, and to keep a database of the available User Agents and their properties. The Agent Personalizer is 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 window area, etc.

### 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. 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 Agents (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 does not necessarily flow through the Mediator Agents. The process of communication is the following:

1. The Network Agent who initiates the conversation gives a required capability list to the local Mediator Agent to find a remote Network 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. MAs may use learning algorithms to solve this problem economically.
3. The initiating agent chooses one from the list of found agents (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 NA is allowed or not,
   - what should be the properties of the communication channel between the two NAs.
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 are to keep a database of available Network Agents, to cooperate with other Mediator Agents in searching desired agents, and to help in building and maintaining communication channels between agents. The first two tasks indicate 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. However the details of this description language are out of the scope of this paper.

### 3.2 Examples of Agents

In the following a few examples are presented to demonstrate how everyday applications can fit into the general framework. All of these examples have running prototypes in the current PHP implementation.

#### 3.2.1 Clock - A Simple User Agent

The agent starts by displaying a clockface and then periodically updates the clockface 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 (clockface 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. The time to display is provided by the system that displays the agent. However, with the use of a Network Agent, the time could be delivered from any other host.

#### 3.2.2 Regular Internet Servers As Simple Network Agents

Regular Internet servers (WWW, Gopher, Finger, etc.) are in fact 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 service can be integrated to our system as a Service

Agent. 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 her needs.

#### 3.2.3 Watchdog - An Application Based On Communicating Network and User Agents

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 that 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 (Figure 6). 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. Finger Agents provide the functionality of a finger daemon. The Watchdog Agent then periodically queries finger information, filters it and displays the results.

### 3.3 Implementational Notes

The current PHP system is implemented in Perl, and agents are implemented in Perl and/or Java. Further directions are to integrate some of the accepted object distribution mechanisms [1] (e.g., CORBA [2], Java RMI) and multi-agent frameworks (e.g., [6]). In this way PHP services could exploit not only the strength of the agent metaphor but also the reliability of these architectures.

## 4. Summary

The research described here aimed at the development of an enhanced working environment on the World Wide Web. The PHP is a working prototype of the results of the presented research. As it was shown through examples it opens a new direction to the application of individually customized dynamic Web pages and thus to more comfortable teleworking environments. The system presented here is based on an agent based general framework for Internet services.

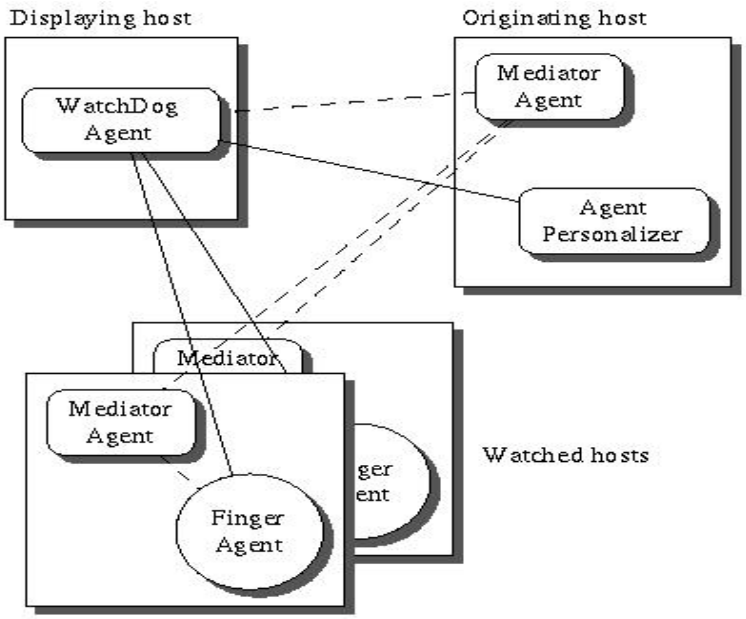


**Figure 6**

## 5. References

[1] M. Bíró, A. Micsik, T. Remzső: *Distributed Object Management, a Survey*, CON '93, Szombathely, Hungary
[2] *The Common Object Request Broker: Architecture and Specification*, Revision 2, Object Management Group
[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. http://www.msci.memphis.edu/~franklin/AgentProg.html
[4] D. Gilbert, Pete Janca: *Intelligent Agents White Paper*, IBM Intelligent Agent Center of Competency, http://www.raleigh.ibm.com/iag/iagwp1.html
[5] L. Kovács: *The GroupSPACE Concept*, Proc. of the 14th IEEE International Conference on Distributed Computing Systems (ICDCS-14), Poznan, Poland, 1994
[6] Java Agent Template, Stanford University, 1997, http://java.stanford.edu/java_agent/html/
[7] H. S. Nwana , *Software Agents: An Overview*, Knowledge Engineering Review, Vol. 11, No 3, pp.1-40, Sept 1996., Cambridge University Press, 1996, http://www.cs.umbc.edu/agents/introduction/ao/
[8] M. Wooldridge, 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. ftp://ftp.elec.qmw.ac.uk/pub/isag/distributed-ai/publications/ECAI94-WS.ps.Z