Single-agent and Multi-agent Approaches to WWW Information Integration

Yasuhiko Kitamura[1], Tomoya Noda[1], and Shoji Tatsumi[1]

Department of Information and Communication Engineering
Faculty of Engineering, Osaka City University
3-3-138 Sugimoto, Sumiyoshi-ku, Osaka 558-8585, Japan
{kitamura, tnoda, tatsumi}@kdel.info.eng.osaka-cu.ac.jp
http://www.kdel.info.eng.osaka-cu.ac.jp

Abstract. The WWW is a most popular service on the Internet and a huge number of WWW information sources are available. Conventionally we access WWW information sources one by one by using a browser, but *WWW information integration* gives a unified view to users by integrating multiple WWW information sources elaborately. In this paper, we introduce our single-agent and multi-agent approaches to WWW information integration.

1 Introduction

As the Internet spreads out over our society, it is becoming one of indispensable social infrastructures which support our daily life. Among various services offered by the Internet, the WWW (World Wide Web) becomes most popular and a huge number of WWW information sources support our research, business, and personal activities. Conventionally we access WWW information sources one by one by using a browser and utilize each of them as an independent information source. On the other hand, *WWW information integration* gives a unified view to users by integrating multiple WWW information sources elaborately.

For instance, the Softbots project led by Oren Etzioni at University of Washington aims at integrating information sources on the Internet to offer more advanced and better quality information services [1]. MetaCrawler[1] is a meta search engine which integrates query results obtained from multiple generic search engines and improves the quality of query results [6]. Ahoy![2] is a special purpose search engine to find a personal homepage by filtering query results from the Metacrawler by using other information sources such as an E-mail database [7].

In academic research fields, WWW information integration plays an important role. In the Human Genome Project, researchers are developing a number of various biological databases, concerning sequences, 3D structures, functions, and bibliography of DNA and protein etc., which are now available through the WWW. These databases will be more useful if they are interoperable. As a first

[1] http://www.metacrawler.com/
[2] http://ahoy.cs.washington.edu:6060/

step to information integration, DBGET[3][3] and SRS[4][2] have been developed to make data items interreferable by using hyperlinks.

The above examples are precursors of WWW information integration but have some drawbacks such that they achieve only page-level integration on third-party servers where users cannot specify how to integrate the information. In the future, we need to achieve WWW information integration more easily, freely, and elaborately on our client machine.

In this paper, we introduce our two approaches to WWW information integration. As our first approach, we developed a single-agent system called MetaCommander. The MetaCommander collects WWW pages, extracts data, and merges them on behalf of a user following a script described by the user. We then, as our second approach, introduce a multi-agent system called Personal WWW Information Integrator (PWII). PWII consists of Information Agents which extract data from WWW servers, Task Agents which synthesize extracted data, and Interface Agents which mediate between the system and users. We combine these agents on a single platform with GUI, hence we can not only integrate WWW information easily and flexibly, but also we can reuse and share the agents among users.

2 MetaCommander

The MetaCommander is an agent with the following features which automatically collects and sorts data from distributed WWW information sources on behalf of its user.

Paragraph-level integration. The purpose of developing MetaCommander is to achieve WWW information integration at paragraph level, while current WWW tools like search engines and bookmarks of WWW browser achieve it at page level. Hence, we can cut and paste paragraphs in various WWW pages at different sites and produce new pages automatically.

Script language. Because current WWW pages (HTML documents) contain little semantic information, cutting and pasting paragraphs in WWW pages is not an easy task for an agent, and its user needs to assist the agent in teaching how to do it precisely. For this purpose, we adopted an easy-to-use script language to represent the user's requirements for integrating information from distributed WWW servers. It is easier to represent the requirement in a script language than a high-level language although it may constrain the functionality. The script should be easy enough for a non-expert of computer, who has a little knowledge of programming, to use. However, for facilitating complicated information integration, the script should have basic functions that ordinary programming languages provide such as local file access, mathematical/logical calculation, control, and so on.

[3] http://www.genome.ad.jp/dbget/dbget.html
[4] http://www.embl-heidelberg.de/srs/srsc

Java implementation. The system should run on a client machine like PC. Hence, we implemented the MetaCommander with the Java language, which is platform independent and runs on most of current platforms like UNIX, MacOS, and Windows.

2.1 System Components

We show the MetaCommander components and the data flow in Fig. 1. We give commands to the MetaCommander by describing a script. The MetaCommander interprets the script and executes it. It uses the HTTP (Hyper Text Transfer Protocol) to access WWW servers through the Internet. It also can read and write local data files. Finally, it outputs an HTML text, and we can display it on an ordinary WWW browser.

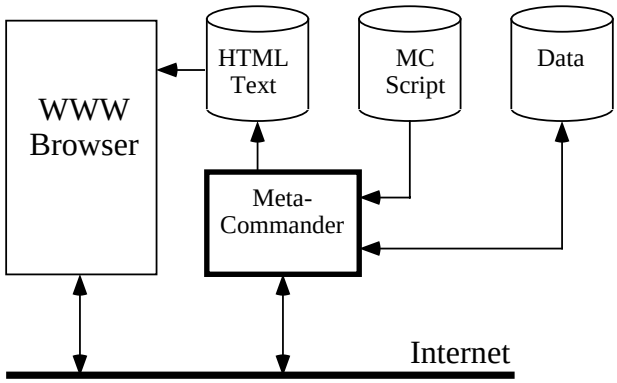

Fig. 1. MetaCommander components.

2.2 MetaCommander Script

We show major functions of MetaCommander script in Table 1.

The style of MetaCommander functions is similar to that of the language C like `function(arg1, arg2, ...)`. The scope where a function is effective is specified by braces('{' and '}'). By adding `else` with braces after the function name, we can specify another scope which is effective when the function fails.

For example,

```
getURL( "http://www.ieice.or.jp/" ) {
    print
} else {
    print( "ERROR" )
}
```

Table 1. Major functions of MetaCommander script.

Group	Name	Description
Page Retrieve	`getURL`	get a page at URL
	`postURL`	get a page at URL by posting to CGI
	`multipartURL`	get a page at URL by posting in MIME format
	`fileURL`	download a page at URL
	`password`	set user's name and password
Data Extraction	`getAnchor`	extract an anchor
	`getString`	extract text strings
	`searchString`	extract a paragraph
	`searchTag`	extract an HTML tag
	`cutString`	extract text strings by specifying the region
Layout	`tag`	insert an HTML tag
Print	`print`	print data to standard output
File Access	`open`	open a file
	`eof`	check if end of file
	`getline`	read a line from a file
	`putline`	write a line to a file
	`fprint`	write data to a file
Variable	`set`	set data to a variable
	`unset`	unset data in a variable
	`calc`	evaluate a mathematical expression
	`strcat`	concatenate variables
	`chop`	remove CR code in variable
Control	`if`	jump on condition
	`foreach`	repeat operations for each data
	`while`	begin a loop
	`exit`	halt
	`break`	break the loop
	`continue`	jump to the top of loop

outputs the contents of the designated page when the MetaCommander succeeds to access it, otherwise it outputs "ERROR."

Unique functions of the MetaCommander are those for WWW page retrieval, data extraction, and layout. Functions for WWW page retrieval collect HTML documents from WWW servers at the designated URLs. The function `getURL` is mainly used with a URL as its argument. When we need to send data to a CGI (Common Gateway Interface) through a form, one of `getURL`, `postURL`, or `multipartURL` is chosen depending on the form type. When we need user's authentication to access a secured server, `password` function is used to specify user's name and password.

Functions for data extraction are used to cut data from and paste them to HTML documents. They are `getAnchor` that extracts link data, `getString` that extracts pure text strings without tags, `searchTag` that extracts a tag,

`searchString` and `cutString` that extract text strings by using key strings. For example,

```
cutString("ABC","XYZ") {
    print
}
```

cuts text strings beginning with "ABC" and terminating with "XYZ."

The `tag` is a function for layout, which inserts a tag in an HTML text. The `print` is a function for outputting a specified data into the standard output (a file named "meta.html"). If the function has no argument, the contents in the valid scope become the output.

Furthermore, functions for local file access, variables, calculation, control, and so on, are prepared. Subroutine calls are also available.

2.3 Applications to Scientific Research Domain

The Human Genome Project is an international project to elucidate all the genetic information and mechanism of human body. This project has a big expectation that the genetic information will contribute to the revolutionary progress in biology, medicine, and pharmacy such as elucidation and treatment of genetic diseases. On the other hand, information processing technologies are essential to manage and utilize a huge amount of various data obtained by biological experiments[9]. At present, the WWW is a nucleus technology to share information among genome researchers in the world, and a number of various databases and analysis tools are open to the researchers such as the GenomeNet[5] in Japan.

To integrate databases by hyperlinks however causes an operational problem. For example, it is possible to collect reference information from a nucleic acid database GenBank by sending a query through its CGI, but the operations become repetitive when a user wants to collect related information from a number of entries because each entry is contained in a single WWW page. For this problem, the MetaCommander can automate to follow hyperlinks and cut some parts (ex. reference data) out of GenBank pages, and integrate them into a page. We here show a MetaCommander script to execute the above operations in Fig. 2.

At first, the URL of GenBank server (Line 1), the number of entries to retrieve (Line 2), and keywords (Line 3) are set to variables. Then, a MetaCommander gets access to the GenBank server (Line 4) and obtains a list of links to entries as shown in Fig. 3(a). It extracts the link (Line 6) and outputs it (Line 8). As the URL designated by the link has been set to a system variable $_, it follows the link (Line 10). Since the server returns a page shown in Fig. 3(b), the MetaCommander cuts reference data between "REFERENCE" and "FEATURES" (Line 11) and outputs them (Line 12). Finally an HTML text shown in Fig. 3(c) is obtained. In Fig. 4, we show an example produced by executing the script in Fig. 2.

[5] http://www.genome.ad.jp

```
1:set( url, "http://www.genome.ad.jp/htbin/www_bfind_sub" )
2:set( max_hit, 5 )
3:set( keywords, "hiv human")

4:getURL($url,"dbkey"="genbank-today", "keywords"=$keywords,
  "mode"="bfind", "max_hit"=$max_hit) {
5:  file( "result.html" ) {
6:    getAnchor($max_hit) {
7:      tag("LI")
8:      print
9:      tag("BR")
10:      getURL($_) {
11:        cutString("REFERENCE","FEATURES",1,0){
12:          tag("PRE") { print }}}}}
```

Fig. 2. A MetaCommander script to collect reference information from GenBank.

2.4 Application to Business Domain

Electronic commerce becomes one of most popular applications of the Internet. There are a number of virtual shops which carry various goods. An advantage of electronic commerce for a consumer is that he/she can compare the prices of goods and choose cheapest ones much easier that he/she does in traditional commerce. The MetaCommander makes the comparison shopping more easier.

Fig. 5 shows a collection of price information about a graphic board for PC named "Revolution 3D" from 3 Japanese virtual shops (System Works[6], DOS/V Paradise[7], and TWO TOP[8]).

Collecting price information from virtual shops is not easy. There is no definitive rule to find a catalogue page which contains an item which a customer like to buy because how to build catalogue pages depends on the shop. Normally a customer is required to follow several links to find the page from the top of virtual shop homepage and sometimes he/she has a difficulty to find it. If we describe the above process in a script, it will be a quite complicated and lengthy one. As a remedy, we can incorporate a use of ordinary search engine into the script. In the script which outputs Fig. 5, we consult a search engine with a keyword "Revolution 3D." Although it returns a number of links concerning the keyword, we can sift the output by using the URLs of virtual shops. For example, if `http://twotop.exa.co.jp/tuhan/video.html` is included in the links from the search engine, we get to know that the TWO TOP (`twotop.exa.co.jp`) carries "Revolution 3D" and that it is located at `http://twotop.exa.co.jp/tuhan/video.html`. By incorporating a search engine, we can reduce the length of script to about 200 lines.

[6] http://www.systemworks.co.jp
[7] http://www.dospara.co.jp
[8] http://twotop.exa.co.jp

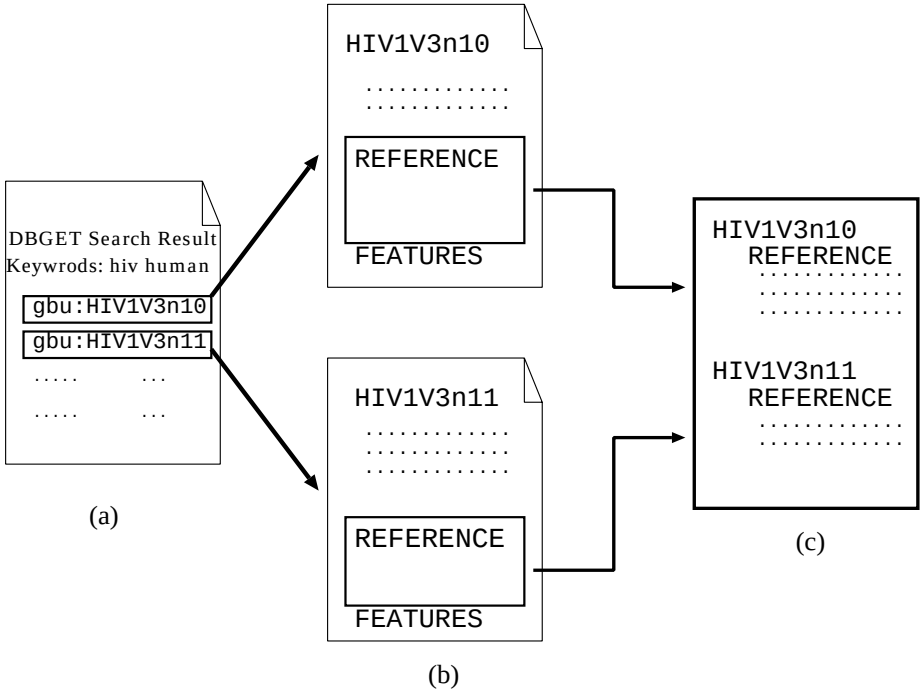

Fig. 3. Collecting reference information from GenBank.

2.5 Future Work

The MetaCommander is now downloadable from our WWW site[9] and has been applied to various purposes. Our future work on the MetaCommander is summarized as follows.

Automatic script generation. The MetaCommander reduces the burden of WWW information integration but developing scripts still remains as a tough work especially for novice users. Now we have interest in automatic script generation based on "demonstration-oriented user interface" (DoUI) [5] which produces a script that mimics the user's operations on a WWW browser.

Sociability. In the future, more automated WWW access tools like the MetaCommander will be widely used, and this will make the traffic in the Internet increase rapidly. In the current Internet environment, selfish users can use the resource (network bandwidth and servers) as much as they like. For this issue, the sociability of agent would work effectively. For example, by monitoring the load of network and servers, agents can selectively access less-loaded servers, or schedule the access plans cooperating with the servers.

[9] http://www.kdel.info.eng.osaka-cu.ac.jp/ mc/

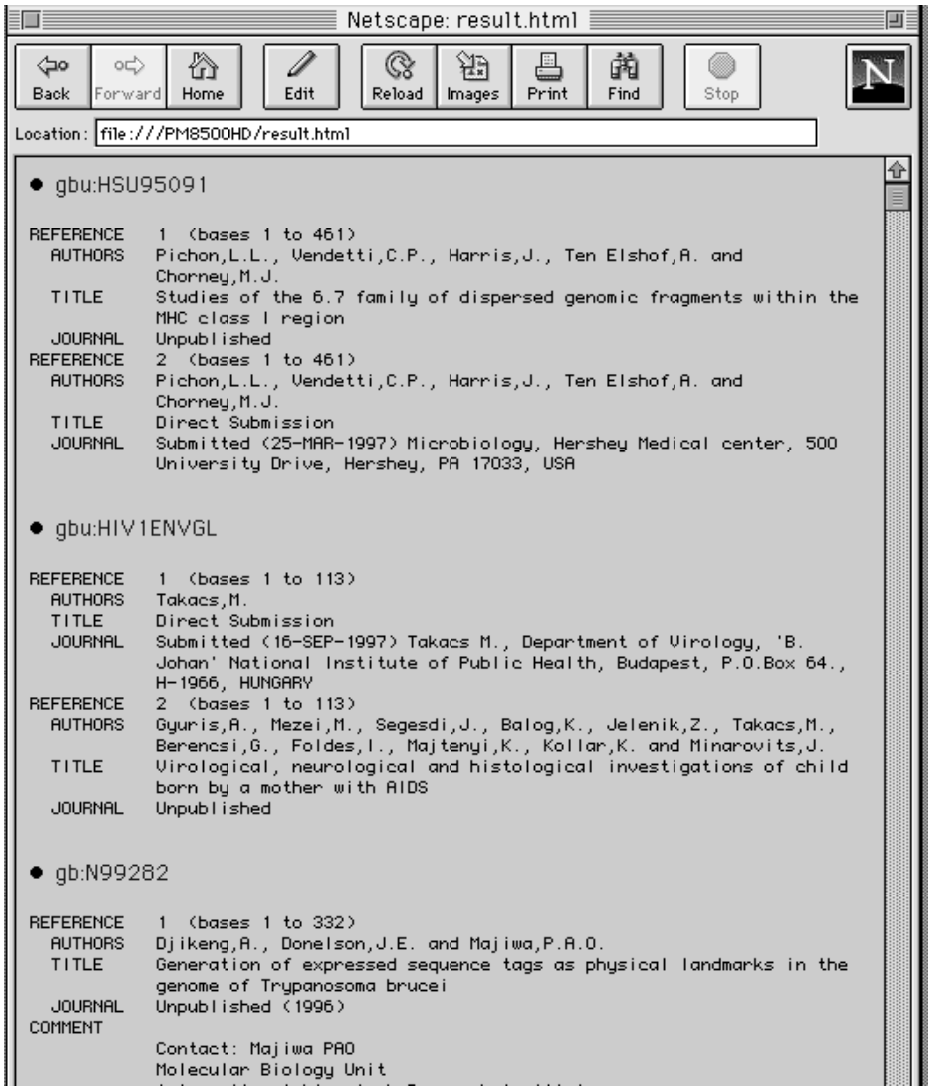

Fig. 4. A WWW page obtained by executing the script in Fig. 2

Fig. 5. Collecting price information from virtual shops. The table consists of price information about a PC graphic board "Revolution 3D" collected from three Japanese virtual shops; SYSTEM WORKS, DOS/V Paradise, and TWO TOP, on the Internet. Each line consists of the name of virtual shop, the specification, and the price in Yen.

3 Personal WWW Information Integrator

The MetaCommander, in which a WWW information integration task has to be described as a script, has following shortcomings.

- When a WWW information integration task is complicated, the script also becomes complicated and difficult to be updated or modified.
- When the structure of WWW information source is updated, it may affect the whole script.
- It is not easy to share or reuse scripts among users.

To deal with the above shortcomings, we are developing a multiagent WWW information integration system called Personal WWW Information Integrator (PWII) where a WWW information integration task is viewed as a cooperative task by multiple agents. PWII has the following features.

- We agentize a WWW server on the Internet as an Information Agent which is interoperable on a client machine. An Information Agent accesses a WWW server, decomposes an obtained HTML document, and transforms it into a structured data with semantic tags.
- We introduce Task Agents which transform and synthesize data extracted by Information Agents and Interface Agents which mediate between a user and the system.
- We introduce a GUI environment in which agents are integrated.
- We make agents shareable among users and agent developers.

3.1 Components of PWII

We show the PWII which consists of Information Agents, Task Agents, and Interface Agents in Fig. 6 [8]. Here we concisely describe each of them as follows.

Information Agent An Information Agent manages accesses to a WWW server on the Internet and gives an interface to other agents. It accesses a WWW server through the HTTP and obtains a HTML document. It then decomposes the document into paragraphs and restructures them with semantic tags. To represent this structured data, we plans to use XML (eXtensible Markup Language). We call the output of Information Agent *message*.

A straight forward method to transform a HTML document to a XML document is by hard coding using script or programming language, but it has drawbacks such that the coding sometimes can be complicated and this method is not robust against the structure change of WWW pages. Hence, we may be able to adopt an approach by [4] which use a template to extract information automatically.

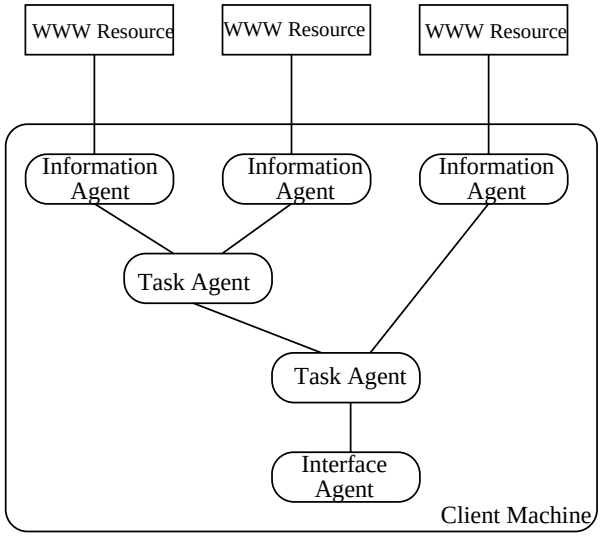

Fig. 6. Components of PWII

Task Agent For WWW information integration to satisfy requests from a user, we need to transform and synthesize messages from Information Agents. Because requests are various and depend on users, a generic task agent, which performs generic data transformation or synthesis such as relational operations of database management system, may be useful since we can satisfy many requests by reusing or replicating the generic agents.

Interface Agent An Interface Agent mediates between its user and other agents. It receives parameters for WWW server access from its user and displays the result to the user or saves it in a file.

3.2 Integration Network

A WWW information integration task is achieved through message exchange among Information Agents, Task Agents, and Interface Agents.

Here we show an integration of search engines as an example scenario of WWW information integration in Fig. 7. We assume query keywords are stored in a file. An Interface Agent (File Input) reads keywords from the file and send them to Information Agents (Search Engines A and B). Search Engine A sends its output to Task Agent (Filtering) for message filtering. The Task Agent passes search results with score of 80% or more to another Task Agent (Union). Search Engine B sends all the results to Union. Union compares both results and takes ones which are included in both results and sends them to another Interface Agent (File Output). The Interface Agent saves the result in a file.

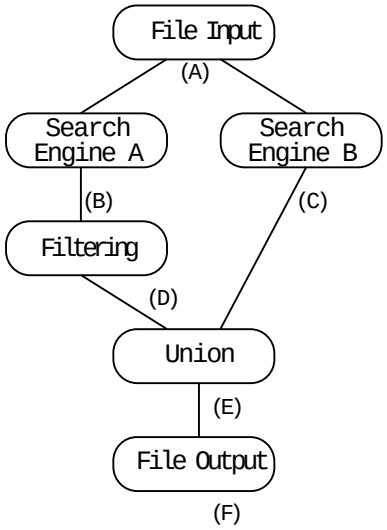

Fig. 7. Integration of Search Engines.

3.3 Agent Integration on GUI

In Fig. 8, we show our GUI system for agent integration. This GUI consists of a canvas and buttons. We locate agents and links on the canvas, and can edit them like a drawing tool. We can use buttons whose functions are shown in Table 2 to manipulate agents and links.

A most common manipulation is as follows. We choose agents from the pull-down menu and locate them on the canvas. We then add links between agents to specify the flow of data among agents. We can change the properties of an agent if needed by using its property window which appears when we click the agent. Fig. 9 shows the property window of search engine called `goo`[10]. The parameters correspond to the original CGI of the search engine and we can change these except KEYWORD which is set by other agent (`input file`) connected by a link because `$KEY` means outer reference. This `goo` agent outputs the number of hits (`HITS`) and the list of URL (`RESULT`) from the search engine and sends them to `filter` agent.

Once we construct an integration network, we can execute the integration task by pressing "VIEW." The execution starts from root agents of the message flow tree. When an agent receives a message, it processes the message and send result messages to other agents as it is specified in the integration network. When a leaf agent is an Interface Agent for file access, the output is saved in a file in the HTML format, so we can see the result by using a WWW browser.

[10] http://www.goo.ne.jp/

When we press "EXPORT," the system generates a MetaCommander script whose operations are the same as that of the integration network on the canvas.

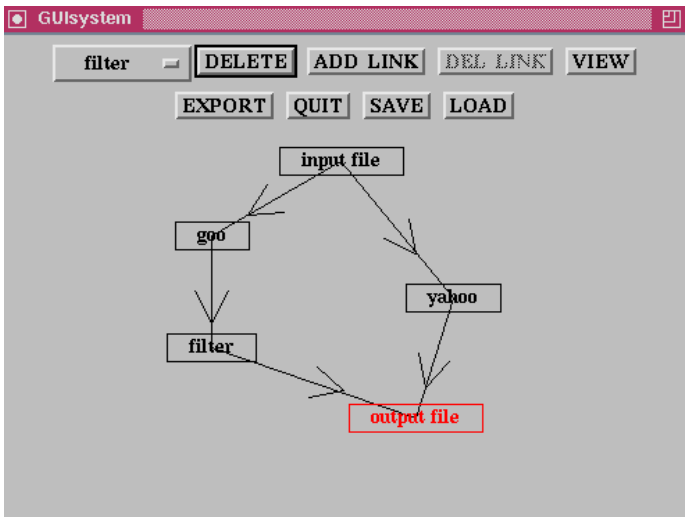

Fig. 8. GUI of Personal WWW Information Integrator.

Table 2. Functions of Buttons

Name	Function
DELETE	delete a specified agent
ADD_LNK	add a link between two spedified agents
DEL_LNK	delete a specified link
VIEW	display the result
EXPORT	output MetaCommander script
QUIT	quit
SAVE	save the integration network
LOAD	load a saved integration network

3.4 Implementation

We developed a PWII prototype as a Java application. In this prototype, an agent is coded as a class, so we can share one among users easily by copying the class from a client machine to another.

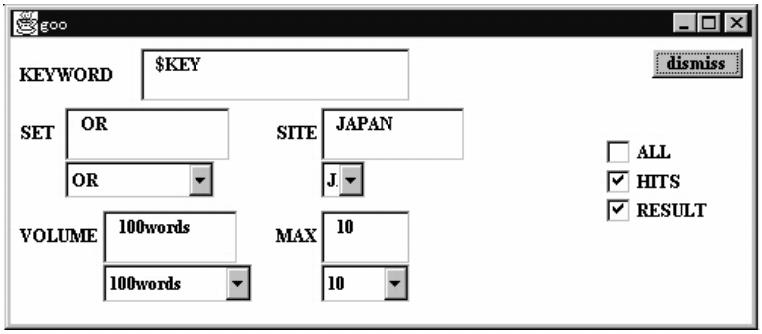

Fig. 9. Property window of goo agent.

4 Summary

In this paper, we introduced two systems MetaCommander and PWII for WWW information integration. Currently, these systems work following a script or an integration network specified by a user. For our future study, we have interest in making these agents autonomous and smart.

Acknowledgements

This work is partly supported by a Grant-in-Aid for Scientific Research on Priority Areas, "Genome Science," from the Ministry of Education, Science, Sports and Culture of Japan.

References

1. Etzioni, O.: Moving up the information food chain: deploying softbots on the World Wide Web. Proc. 13th National Conference on Artificial Intelligence (1996) 1322–1326
2. Etzold, T., Argos, P.: SRS – an indexing and retrieval tool for flat file data libraries. CABIOS, **9**(1) (1993) 49–57
3. Fujibuchi, W., Goto, S., Migimatsu, H., Uchiyama, I., Ogiwara, A., Akiyama, Y., Kanehisa, M.: DBGET/LinkDB: an Integrated Database Retrieval System. Pacific Symp. Biocomputing **3** (1997) 683-694
4. Hsu, J. and Yih, W.: Template-based information mining from HTML documents. Proceedings of 14th National Conference on Artificial Intelligence (1997) 256–262
5. Knoblock, C.A. et al.: Modeling web sources for information integration. Proc. 14th National Conference on Artificial Intelligence (1997) 211–218
6. Selberg, E., Etzioni, O.: The MetaCrawler architecture for resource aggregation on the web. IEEE Expert, **12**(1) (1997) 11–14

7. Shakes, J., Langheinrich, M., Etzioni, O.: Dynamic reference sifting: a case study in the homepage domain. Proceedings of 6th International World Wide Web Conference (1997)
8. Sycara, K., Pannu, A., Williamson, M., Zeng, D., Decker, K.: Distributed Intelligent Agents. IEEE Expert, **11**(6) (1996) 36–45
9. Takagi, T.: Application of deductive databases to genome informatics. Journal of Japanese Society for Artificial Intelligence **10**(1) (1995) 17–23 (in Japanese)