Agent-based Adaptive Selection and Interaction to Z39.50 Servers

Camino Fernández Llamas Ignacio Aedo Cuevas Paloma Díaz Pérez
Universidad Carlos III de Madrid
DEI - Laboratory
Av. Universidad 30, E-28911 Spain
{cfl,aedo,pdp}@inf.uc3m.es

Vicente Matellán Olivera
Universidad Rey Juan Carlos
Systems and Communications Group (GSyC)
C/ Tulipán s/n. Móstoles E-28933 Spain
vmo@gsyc.escet.urjc.es

Abstract

Digital libraries and OPACs are offering an enormous collection of information resources which can be accessed through Internet by quite different users thanks to the existence of standard protocols for information retrieval such as Z39.50. However, some users may find difficult to get the items they are looking for in such a huge bulk of heterogeneous resources since they have to know where and how they have to search for. Thus, one of the main reasons for the disregard of the Z39.50 service has traditionally been the absence of appropriate interfaces and the cost of making alternative searches. WAY-Z39.50 *tries to help spreading the use of these servers by providing easy web access to them, as well as, by facilitating the access to multiple servers at once. The prototype described implements the* WAY *general model based on an agent-based architecture that provides adaptive searching in different servers through an adaptive interface.*

Keywords: *Z39.50, adaptive information access, adaptive interface, adaptive searches, agent technologies, Java*

1 Introduction

Digital libraries and OPACs (*Online Public Access Catalogue*) are offering an enormous collection of information resources which can be accessed through Internet by quite different users thanks to the existence of standard protocols for information retrieval such as Z39.50 [19]. Z.39.50 is a session-oriented and stateful network protocol whose development began in the early 80's. It has evolved from its first version, Z39.50-1988, that was the basis of the WAIS [5] (Wide Area Information Systems) protocol, to the current one: Z39.50-1995 (Version 3). This protocol provides a unified access to structured information catalogs (mainly libraries' catalogs) and supports sophisticated searches in the catalogs.

However, some users may find difficult to get the items they are looking for in such a huge bulk of heterogeneous resources. First, users have to decide which ones are the most appropriate information servers for their queries and, therefore, they need to know which are the available choices for every particular server. Second, users have to deal with different interfaces, and even though the idea of providing a unified interface widely accessible, as the web is, has been a goal in the Z39.50 community [12], the interfaces to these servers have not been really accessible [18] or easy to use. Indeed, different servers provide different search fields and they also may use different languages to interact with the user. In order to solve these problems, several approaches can be taken dealing with issues as multilinguality [8] and the use of adaptivity [21].

In this context, a web-browser accessible prototype named WAY-Z39.50 of adaptive system to access Z39.50 servers is presented. WAY-Z39.50 is an implementation of WAY model [9], which is a general model of architecture whose main goal is to improve information access systems [4] through the fulfillment of three tasks: adaptation of the interface to the user, adaptation of the searching process and extraction and maintenance of updated information about the available information servers.

Next section describes the software architecture of WAY-Z39.50, making special emphasis on design an implemen-

tation issues. The last section describes the evaluation process of the prototype implemented, as well as the main conclusions obtained.

2 Providing adaptive access to information through WAY-Z39.50

WAY-Z39.50 is a particular implementation of a general model for adaptive information access called WAY. Thus, before describing the prototype we will introduce the main features of the model.

2.1 The WAY model

WAY is a general model for adaptive information access that is based on a net of intelligent agents whose mail goal is to make easier both the searching and the interaction processes. On the one hand, given a user's query on a specific server, the agents have to take into account the information about previous users sessions to decide which are the best alternative servers for that query and to perform parallel searches to that servers. On the other, user interactions are kept and analyzed to improve the interface adapting it to the preferences and needs of each user.

The term "agent" in WAY terminology has to be understood in the same way as the Carl Hewitt ACTORS [1], that is, computer programs (objects in actors terminology) capable of concurrent execution, capable of keeping their internal state from one execution to another (persistence), and that show interactive behavior both with humans and with other agents by using messages; augmented by Henry Lieberman definition of the capabilities desired for agents [15], and that can be summarized in adaptivity.

Agents defined in WAY model both execute in the same machine, or in different machines, depending on the user location. In case of remote location of users, which is the normal situation, the agents will use network infrastructure to interchange messages. In particular, WAY-Z39.50agents have been implemented from scratch, that is, without using any commercially available libraries for these kind of software, in Java, and take advantage of Java-RMI facilities for their intercommunication.

The model organizes the agents into three modules (see Figure 1), which are in charge of each of the system tasks: the interaction with the user, the communication with the servers and the acquisition, verification and maintenance of updated information about the location and services of available information servers. These modules are explained below.

1. The **user interaction with the interface** comprises the communication with the user and the interface adaptation. These tasks are performed by two agents:
   - *The user communication agent* that builds and manages the interface depending on information about the user and her environment such as her location, the hardware and software platform being used, etc. and also the information provided by the interface adaptation agent.
   - *The interface adaptation agent* that analyses the user's actions to decide which interface modifications to propose. For instance, the fields used to make queries can be used to suggest a change in the order of the searching fields beginning with those most frequently used.
2. The **user interaction with the searching process** adapts the searching process by assigning every server a mark for every user, depending on the server performance and the response of the user to the previous results obtained from that server, generates parallel searches adapted to the user and manages the communication with the information servers. These tasks are performed by two agents:
   - *The searching collaboration agent* that produces the list of the preferred servers for every user, generates parallel searches on the more appropriate servers for that user (the ones more visited, with best results, with best performance, etc). It also decides how many parallel searches to launch depending on several parameters, such as the machine load or the response of the user to previous offered alternative searches.
   - *The information retrieval agent* that implements the client part of the information retrieval protocol (for instance, Z39.50 in the WAY-Z39.50 prototype). It is in charge of the communication with this kind of servers to obtain both the results of the user's queries and additional information concerning information servers services.
3. The **information about servers** is obtained, verified and maintained by:
   - *The servers searching agent* that looks for information sources, as web pages or news groups, and consults them searching for possible information servers locations that the information servers maintenance agent will try to verify.
   - *The servers information maintenance agent* that receives information about possible servers location and verifies their performances. It repeats the operation with a higher or lower frequency depending on the amount of changes in the data obtained.

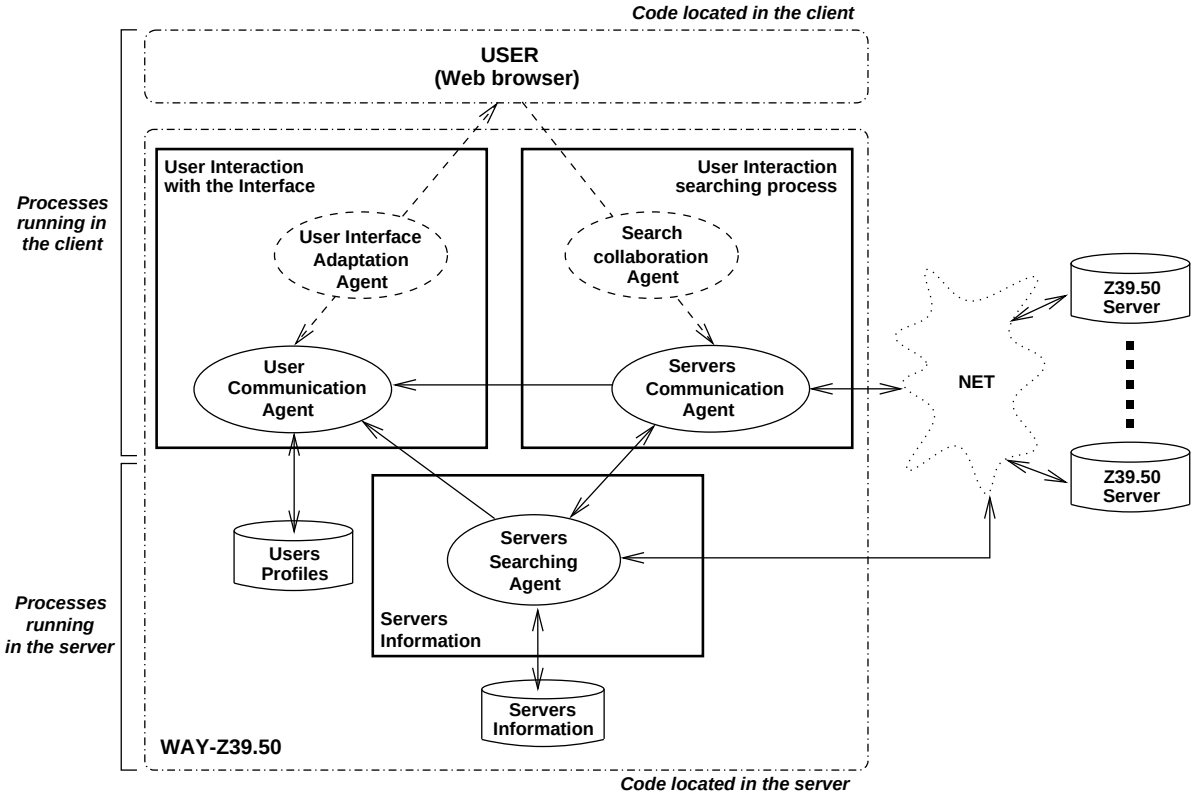

Figure 1. The WAY model

The first implementation of the WAY model is WAY-Z39.50 which provides adaptive access to Z39.50 servers.

There are few other approaches trying to make easier the process of accessing libraries catalogs through Z39.50 protocol. The commercial products are focused on other problems. For instance, the CARLweb system from *CARL Corporation* [1] provides a customizable version of the traditional OPAC; the Millenium system from III (*Innovative Interfaces Inc.*) [2]) is focused on scalability; others are interested on integration of different information sources, on the internal design for its easy adaptation to existing catalogs, etc.

In this point, it is important to distinguish between adaptable systems (i.e. Millenium, CARLweb), and adaptive systems (WAY-Z39.50). Systems that allow the user to change certain system parameters, and adapt their behavior accordingly, are called adaptable. Systems that adapt to users automatically based on their assumptions about them are called adaptive.

Adaptivity in information access can still be considered to be in the research stage. The majority of projects on adaptivity in information access are focused on web browsing assistants [17] as Letizia [14], FAB[3], or Butterfly [16]. However, the application of these techniques to the structured catalogues access domain is still scarce, mainly because of the lack of standardization in the access protocols (the equivalent to HTTP for web). In this way, projects as EUROPAGETE[3] [7] that intends to provide software tools through which users can access ANSI Z39.50 and ISO SR servers, funded by European Union [4] Telematics Libraries Programme are spreading the use of Z39.50

Projects whose goals can be considered close to WAY-Z39.50 are very few. OASIS (Otlet's Adaptive Search Information Service) [6], is a long-term research program funded by Berkeley's Digital Library Project, exploring how to improve the retrieval process by providing simultaneous searches to multiple databases, and combining diverse retrieval results to provide narrow result sets. However, OASIS was fit to MELVYL online catalog, instead of using Z39.50.

Similar goals has BOPAC2 [2], a project funded by the British Library Research and Innovation Center, which provides a World Wide Web front-end that allows simultaneous access to different library catalogues, in this case via Z39.50. The results obtained are clustered together and retrievals may be sorted in a number of way by different

[1] http://www.carl.org
[2] http://www.iii.com
[3] http://europagate.dtv.dk
[4] http://www.cordis.lu/libraries/

criteria. The main disadvantage when compared to WAY-Z39.50 is that the clustering, that is, the organization of the results, and their presentation is made in a predefined and unchangeable way.

2.2 WAY-Z39.50 software architecture

Since the main goal of the WAY model is to make easier for the user to access information, the applications based on the model should themselves offer, as long as possible, the easiest way to be accessed. In order to accomplish this feature, WAY-Z39.50 interface is based on a web page.

However, a static interface is not enough to support the complexity of the Z39.50 protocol. One of the main differences between HTTP and Z39.50 protocol is that the first one is stateless, whereas the searches in the second one are kept in the servers while the connection keeps opened. This makes it hard to deal with an HTML-based system, specially if searches in multiple servers are required. One option could be the use of *cookies* to emulate state, for instance when making parallel searches. However, other features required in WAY specification, such as the interface adaptation, cannot be managed (or hardly can be) by cookies, because they require the server to manage the whole adaptation process: users' actions tracking, evaluation of their actions, modifications in the interface, etc.

In order to deal with this complexity a Java implementation of the WAY architecture was chosen to implement the WAY-Z39.50 system. Using this web based Java powered approach, some important goals are achieved:

- the environment is already familiar to the user, undoubtly web browsers are probable the most widely used software products nowadays,
- the user needs to install nothing new in his/her computer (apart from the web browser) but connecting to a web site, and
- the execution of the application takes place mainly at the user's computer although no software has to be locally stored.

The last one greatly influences the design of the software architecture of the platform. In fact, forcing it to be distributed. One part of the application will be running in the client computer (as a Java applet), and another part will be running in the WAY-Z39.50 server.

In Figure 2, the distributed characteristics of the application are shown. Some of the agents described in the previous section are transfered via HTTP (in form of a Java applet) to the user's computer. One instance of each of these agents exists for each user. These agents are: the user communication agent, the interface adaptation agent, the searching collaboration agent, and the information retrieval agent. The other two agents are kept in the server and there are just one instance of each one for the whole system, as they are not related to the interaction of any particular user, but they provide a service for all the other agents.

The four agents running in the client machine, neither have access to local storing facilities, nor have access to the communication resources. The reason is that, in order to protect web users from dangerous programs, the usual implementation of the Java virtual machine made for web browsers only allows Java applets to establish connections with the machine where their code came from, and do not let them access the local disk (except for leaving cookies).

The first problem is where to store the information that the agents need, as for example their state, the user preferences, etc. Due to the fact that there are different kinds of information and that the information has to be accessed in a structured way, it was decided that a SQL database, as shown in Figure 2, will be the best option to keep the data easily accessible. The database is accessed via Java-RMI (Remote Method Invocation) and JDBC (Java Data Base Connection).

The second problem is how to access the Z39.50 servers, how to make the queries, and how to retrieve the results to the client. In this case, the answer was again the use of Java-RMI, the distributed part of Java programming language. This way, agents running on the user's machine could access the server side, and the server side could access the Z39.50-servers and give the user the data back. This solution offered also the possibility of storing the user session in order to obtain valuable data for the evaluation phase of the prototype.

This means that agents need to ask WAY-Z39.50 for those resources: communication and storage. This makes the WAY-Z39.50 server the key element of the system. Due to the different kind of services to be provided, the server itself has been decomposed into three "servers", each of them in charge of a different set of tasks. The internal structure of the WAY-Z39.50 server, as well as the relationship with the agents of the system are shown in Figure 3.

These relations among WAY-Z39.50 agents and the servers can be shown in the figure and can be summarized into:

1. Both the retrieval and the searching agent use files. Although this is not compulsory, it has been added as an optimization for the temporal storage of the results of previous searches, so that they do not have to be repeated when they are revisited.
2. All agents, except for the retrieval one, need to store some kind of information into the database. Retrieval agent is just in charge of managing Z39.50 connections, using files to temporarily store the results obtained.

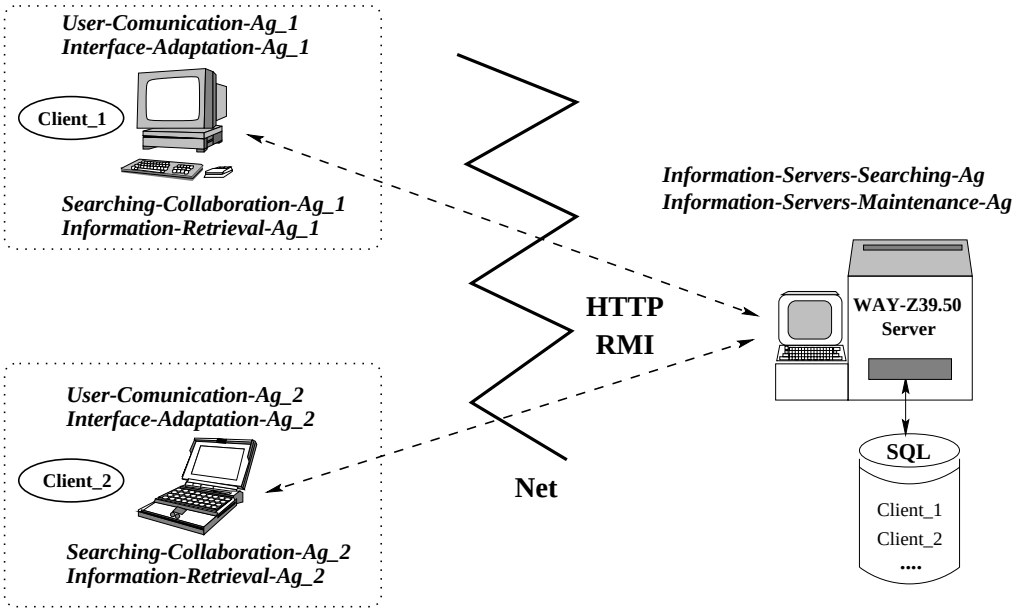

Figure 2. WAY-Z39.50 **software architecture**

3. The interface related agents (user communication and interface ones) do not use Z39.50 service directly. The other agents, for one or another reason, use those services, so there are lines connecting them to the Z39.50 clients server in the Figure 3.

Functions and goals of WAY-Z39.50 agents have been described in the previous section. Their implementation has been made attending those requirements, using Java as the programming language, and fuzzy rules [22] as the adaptation mechanism, both in the interface modifications, in the information searching process, and in the classification of the Z39.50 servers available. This means, that there is a set of human designed rules for each of these functions that defines each agent behavior.

The three servers which comprise the WAY-Z39.50 server were not described in the WAY model section, since they are application dependent. These servers are:

Files server: Its mission is to manage the auxiliary files of the system that are generated by the agents. Those files can be grouped into two categories, the first one is made by the temporal files generated by the agents running in the client machines. For instance, one file is generated for each search made in a server by a user. In addition to this user-generated search, the *searching collaboration agent* will generate parallel searches in the Z39.50 servers, that would result into new files. These files will be used to speed up the process of changing from one server to another one, to refine searches, etc. All the files generated for a user will be removed when that user had finished the session.

The second category of files kept in the server are the log files one. In order to evaluate the system performance, as well as to correlate the user answers to the tests made to evaluate the system; the actions of the user can be stored in a log file. In these log-files every action of the user, as well as every decision made by the agents, are stored.

There is a last category of files managed by the server, the configuration files for the decision making mechanism used by the agents to reason about the adaptation process, the classification of the servers, etc. These decisions are taken by a fuzzy inference motor, also implemented in Java. The difference among the agents are the rules that guide the decision process, and the input and output variables. The definition of these rules and variables are kept in independent files (one for each type of agent) to make their modification easier.

Data base server: is in charge of controlling the access to the SQL database. The database used is PostgresSQL [5], an open-source object relational DBMS (Data Base Managing System) supporting almost all SQL constructs. There are many different informations stored in this database, as for instance the *state* of the agents running in the client machine, each user's preferences (language, colors, preferred servers, etc.), and the information about servers (localization, port, services, speed, timetable, etc.).

[5] http://www.postgreSQL.org

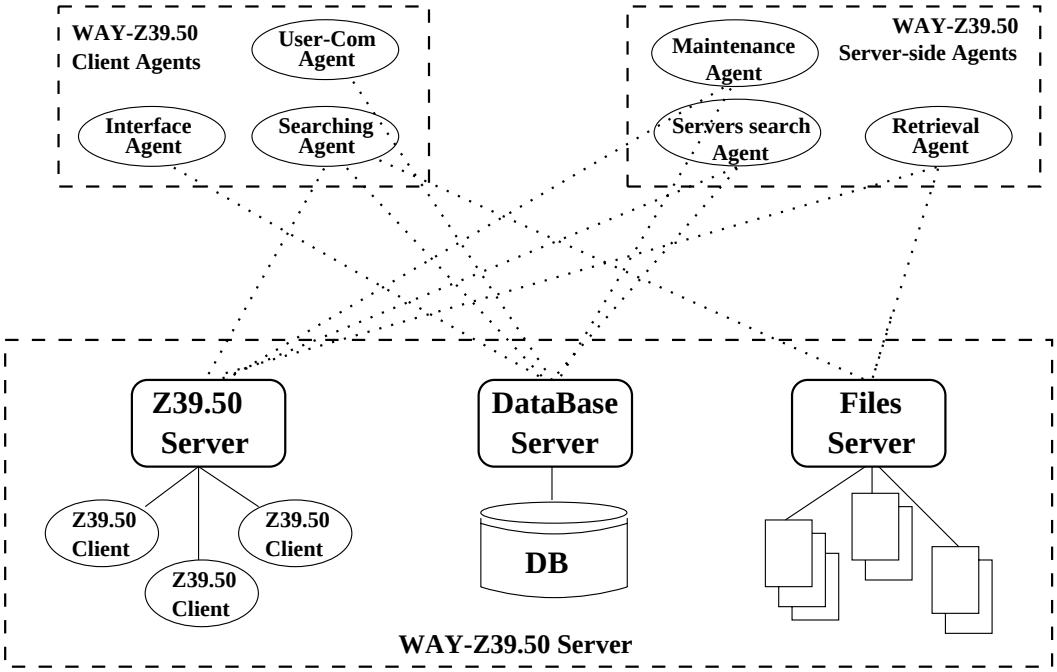

Figure 3. WAY-Z39.50 **server architecture**

The agents, both the local ones (the information servers searching agent and the information servers maintenance agent) and the remote ones (the interface adaptation agent and the searching agent) use JDBC to access the database, using also Java-RMI for the remote invocation of the methods. The access to the objects is synchronized using the Java facilities for multithreading, to avoid problems among agents when accessing the same information.

Z39.50 server: creates objects that implement the client part of the Z39.50 protocol upon request by the agents running in the client machine. For instance, when the user fills the searching form, the information retrieval agent needs to open a Z39.50 connection to the selected server. This agent asks the Z39.50 client server to create an object capable of managing a Z39.50 connection to the server. A reference to this object is returned to the agent, who will manage the communication. In the same way, the searching collaboration agent will ask the Z39.50 server for additional objects, one for each parallel search.

The objects created by the Z39.50 server are managed by the agent that asks for their creation through Java-RMI. This means that the agent will consider these objects as internal structures. The objects will be destroyed when the agent asks for, or when the user leaves the application. While active, the object will keep the state of the connection, which means that the agent itself keeps the state of the connection and can use it to refine searches, access results, sort the results, etc.

The design of the software developed for the WAY-Z39.50 prototype was made using design patterns [11] as the main philosophy, UML [13] as the basic design tool, and reutilisation as the major design goal (in order to make easier the modification of the system according to suggestions made in the evaluation process). The resulting implementation of WAY-Z39.50 prototype is an object-oriented distributed system written in Java (using Java Development Kit v. 1.1.5) made up by around 30,000 Java lines, organized in the following packages:

Patterns: This package is an auxiliary one for the construction of the elements of WAY-Z39.50 prototype. Some of the patterns implemented are the iterator, the observer, etc. The number of lines is around 2,500.

Interface: In this package, all the graphic elements of the system interface are implemented. These elements are used by the interface communication agent to build the user interface. The total number of lines is around 15.000 in the current version of this package.

Agents: This package contains the implementation of the agents that perform most of the system tasks. One of the main sub-packages is the one that implements the *mind* of the agents. Their mind has been implemented

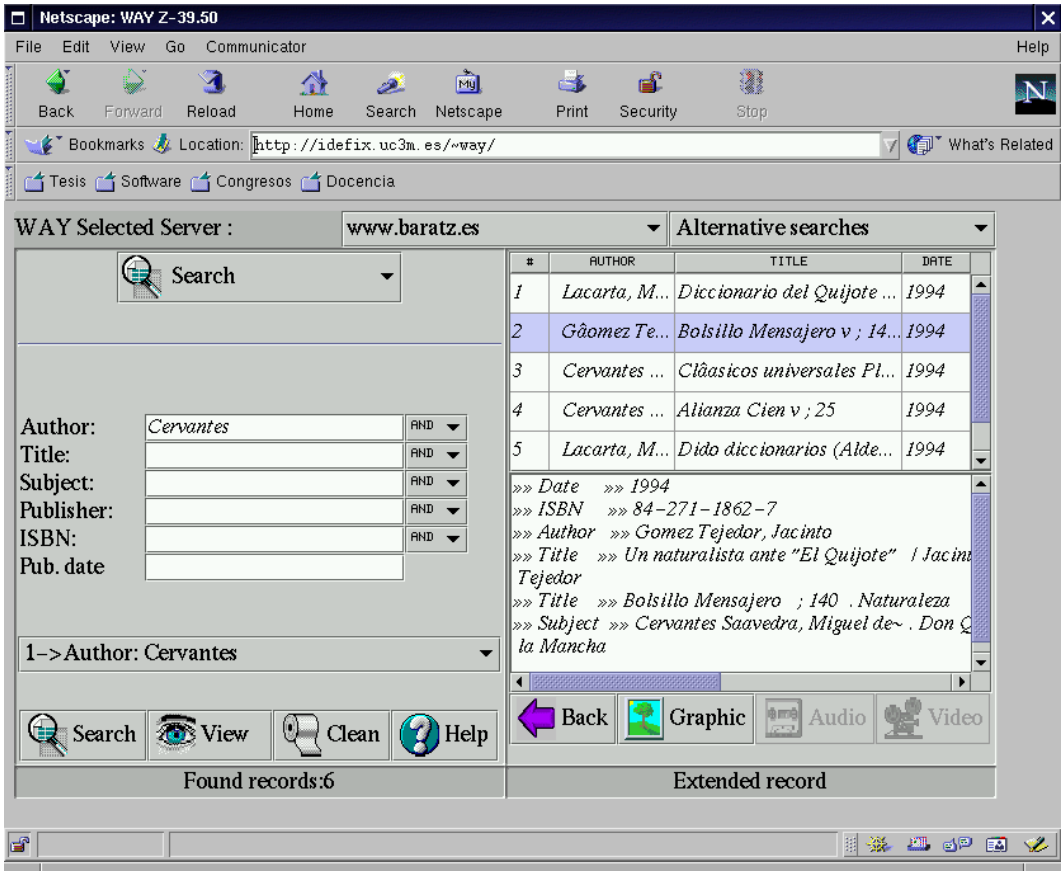

Figure 4. A WAY-Z39.50 **snapshot**

using fuzzy logic, see [10] for a detailed description of this issue. The number of code lines in this subpackage is 3,000, in the whole package there are 8,000 lines.

Z39.50: The primitives that the Z39.50 protocol specifies for the client side of a Z39.50 client are included in this package, counting around 5,000 lines.

3 Evaluation and conclusions

WAY-Z39.50 is a prototype built chiefly to demonstrate that the WAY general model is implementable as well as to assess the model properties. WAY-Z39.50 prototype is still in a Beta phase, however it can already be used and tested in `http://way.uc3m.es` under request. Figure 4 shows the current aspect of the prototype.

In this way, the evaluation of the model has been one of the major concerns since the beginning of the WAY project. The evaluation of the WAY-Z39.50 prototype has been divided into three stages:

1.- Evaluation of the interface: This phase consisted in the evaluation of the default interface, previous to any adaptation. The goal of this test was to check the interface, to identify which aspects of the interface were susceptible of adaptation to the user, etc. The evaluation was made as a test for an homogeneous group of 20 people, using a non-adaptive version of the prototype, during the 1999 autumn.

The interface evaluation phase was organized as a "guided" session with the interface module of WAY-Z39.50 (without the adaptation facilities). "Guided" means that some basic tasks, as locating books written by some authors, refinements of the searches, change of server,..., were assigned to the evaluators. All these tasks have to be accomplished through WAY-Z39.50 graphic and text interface. The evaluation itself consisted on filling a questionnaire about the evaluator opinion about the interface.

2.- Evaluation of the architecture: Consisted in the analytical evaluation [20] of the architectural principles of WAY model. It was implemented as an open test where a set of experts of different areas (librarians, user interface designers, distributed systems architects,

etc.) were asked to assess the model features using its formal specification. This phase was performed in November 1999 and it is still being processed. Anyway, the first results can be anticipated as globally positives.

3.- Evaluation of the prototype: The major concern of the last evaluation phase is to check the viability of the proposed model, this means checking whether WAY can be implemented and helps designers. The best way of checking this is by performing empirical evaluation [20] of an implementation as WAY-Z39.50. The evaluation will be organized as the interface evaluation in order to compare it with the interface evaluation. This evaluation is currently being made.

In the evaluation of the prototype, performance requirements are not the primary concern. However, they can clearly influence evaluators opinions. In order to avoid this kind of biased opinions, the evaluation questions will be focused in the adaptive part rather than in the speed.

In summary, the results obtained from the evaluations made till now show that one of the main advantages that the evaluators have found is the adaptation to the user preferences. The open question currently is whether this adaptation will be supervised or not supervised, that is, if it will be made with or without consulting the user. In the current version, all modifications in the user interface, for instance the order in searching fields, are made in a not supervised mode. According to the evaluator responses it will be considered which adaptations are made with or without user supervision by default, as the user can always change this behaviour.

The remaining work in the evaluation phase is just finishing the users interviews of the third phase and completing the analysis of the data obtained. Anyway, it is already possible to anticipate that both the evaluation of the model, and the evaluation of the prototype consider them as useful, viable and usable.

References

[1] G. Agha. *ACTORS: A model of concurrent computation in distributed systems*. MIT Press, 1986.

[2] F. Ayres and M. Ridley. BOPAC2: a new dimension in OPAC display. *VINE*, (114):50–55, 1999.

[3] M. Balabanovic. An adaptive web page recommendation service. In *Proceedings of the First International Conference on Autonomous Agents*, Marina del Rey, CA, feb 1997.

[4] D. Benyon and P. Palanque. *Critical Issues in User Interface System Engineering*. Springer-Verlag, 1996.

[5] B.Kahle and A. Medlar. An information system for corparate users: Wide area information servers. Technical Report TMC-199, Thinking Machines, April 1991.

[6] M. K. Buckland and C. Plaunt. Selecting libraries, selecting documents, selecting data. In *Proceedings of the International Symposium on Research, Development and Practice in Digital Libraries*, pages 18–21, 1997.

[7] S. . Ciardhuáin and M. Sandfær. The EUROPAGATE project. *VINE*, (97), 1994.

[8] P. Díaz, I. Aedo, C. Fernández, A. Plaza, A. Ribagorda, and C. Díez-Hoyo. Multilingual tools for accessing a spanish library catalogue. *Libri*, December 1997.

[9] C. Fernández, P. Díaz, and I. Aedo. WAY: An architecture for user adapted access to z39.50 servers based on intelligent agents. In *Proceedings of the European Conference on Digital Libraries 98*, 1998.

[10] C. Fernández, V. Matellán, P. Díaz, and I. Aedo. Using fuzzy logic to implement adaptability in WAY-Z39.50. In *Proceedings of the 1999 EUSFLAT-ESTYLF Joint Conference*, pages 291–294, September 1999.

[11] E. Gamma. *Design Patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995.

[12] S. Hammer and J. Favaro. Z39.50 and the world wide web. *D-Lib Magazine*, March 1996.

[13] C. Larman. *Applying UML and patterns: an introduction to object-oriented analysis and design*. Prentice Hall, 1998.

[14] H. Lieberman. Letizia: An agent that assists web browsing. In *International Joint Conference on Artificial Intelligence*, Montreal, August 1995.

[15] H. Lieberman. Autonomous interface agents. In *ACM Conference on Human-Computer Interface*, Atlanta, March 1997.

[16] H. Lieberman, N. V. Dyke, and P. Maes. Butterfly: A conversation-finding agent for internet relay chat. In *International Conference on Intelligent User Interfaces*, Los Angeles, January 1999.

[17] P. Maes. Reflections of agents that reduce work and information overloa d. In M. T. Maybury and W. Wahlster, editors, *Readings in Intelligent User Interfaces*. Morgan Kaufmann Press, 1998.

[18] P. Marshall. WAIS: The wide area information server or anonymous what? *Computing and Communications Services*, 1992.

[19] J. J. Michael and M. Hinnebusch. *From A to Z39.50: A Networking Primer*. Mecklermedia, 1995.

[20] J. Rubin. *Handbook of Usability Testing*. John Wiley & Sons. Inc., 1995.

[21] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing Company, third edition edition, 1998.

[22] H.-J. Zimmermann. *Fuzzy Sets. Theory and its Application*. Kluwer Academic Publishers, Boston, MA (USA), 1990.