INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

WEB BASED CINDI SYSTEM

Graphical User Interface Design and Implementation

XIAOMEI YANG

A Major Report

In

Department of Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

August, 2001

© XIAOMEI YANG, 2001

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-64089-2

Canada

Abstract

Web Based CINDI System

Graphical User Interface Design and Implementation

Xiaomei Yang

As the World Wide Web grows, more and more information will be available on the Internet. However, how to organize such information and let people search for it efficiently is a big problem. As we know, there are a lot of search engines on the web. However, because of the lack of a standard indexing scheme and an informative query interface, they still cannot offer the Internet users a very efficient way to search for information. The CINDI (Concordia Indexing and Discovery) system provides a standard index scheme called Semantic Header, which is introduced by Dr. Bipin C. Desai et al. Moreover, the CINDI system provides an informative query interface for users and a resource registration interface for contributors.

This report presents the design and implementation of the CINDI system based on the World Wide Web. It emphasizes on the graphical user interface (GUI) design and implementation for the CINDI system. UML (Unified Modeling Language) has been used in the design and analysis procedure of CINDI system. PHP has been used in the implementation of the user interface. MySql has been used to implement the database of the system. The Apache web server allows the clients' request and stores the useful information from the database by using HTTP protocol.

Acknowledgement

First of all, I would like to thank my supervisor, Dr. Bipin C. DESAI for his encouragement and valuable suggestions. This work would not what it is today.

Moreover, I would also like to thank Wen Tian, as member of the CINDI project, give me many good ideas during the valuable discussions.

Finally, I would like to dedicate this work to my parents, for their support and encouragement during my master program study.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview of the Exiting Virtual Library

Because of the rapid growth of the Internet, more and more information sources are available on the World Wide Web. This information includes texts, computer programs, books, electronic journals, newspapers, organizational directories, as well as local and national directories of various types, sound and voice recordings, images, video clips, scientific data, and private information services such as price lists and quotations, databases of products and services, and specialty newsletters. [BCD 95] To organize this information on the web, virtual libraries (or digital libraries) have been built.

Virtual libraries are social as well as technological entities. Their purpose is to help people do knowledge work, so that knowledge processes can be carried across space and time. Virtual libraries are designed, used, and evaluated in a context of work and community. Virtual libraries then interact with this context, changing and being changed by it. An effective virtual library must be designed and evaluated with sensitivity as to how knowledge is created and understood, and as to how work is done, in a context of knowledge communities, which share practices and tools. Also, it requires proper information indexing. Secondary information, called meta-information, must be extracted

and used as an index to the available primary resource. Moreover, it needs an automated search system that allows easy search for and access to relevant resources available in the library. [BISO 00]

1.1.1 The Problem of Existing Virtual Library

In the past few years, many such search systems have been developed and some of them have gained wide acceptance on the Internet. These include Alta Vista, Yahoo, Lycos, Google, HotBot etc. Some of these are manually generated indices while others are generated by robots. [KOSB 96] Some of these robot-based systems also allow manual index entry. Moreover, most of these are specialized for the WWW. For example, the most popular search system Google collects the information and catalogs it by its own indexing system. Users can input any keywords they want to search the documents they require. This scheme really lacks standardization. Since different people have different perspectives on the same single concept, the same document may be indexed into different catalogues in different indexing systems. As a result, users of the system may not find the information they need in the catalogue they think the document belongs to. Since the system searches the information by the words the users input, it can cause lots of miss-hits since same words may have different meanings in different domain and different place. For example, in Google when you search "PHP", the system can return 21,000,000 records to you. However, many of these are documents about MySql with the word "PHP" appearing in them.

All in all, these existing search systems have the following drawbacks:

- Lack of standard: Different virtual libraries may describe the same resource in different catalogue.
- Lack of effective search: Many systems cannot search the exact information that users want. A user often has to retrieve the pertinent source from thousands of search result.
- Lack of consistency: Many systems just have the link to the resource, but do not store the resource. Once the resource storage has been changed, users cannot find the resource from the original link.
- Lack of customizing support: Most systems offer little support for customizing how they handle different information formats and index/search schemes.

The creation of indices based on search robots, worms, spiders, or crawlers has the following disadvantages [BCD 94]:

- Repeated attempts by robots to find new resources increases the traffic on the network. The number of these robots is increasing, and system administrators may disallow visits by robots.
- A robot-based approach will become difficult to justify if the network switches to a fee-for-use mode of operation.
- The type of data gathered by a robot is not useful because it is too simple to support discovery. This is the case in spite of the fact that such indices, for the lack of a better tool, have been useful to date.
- Natural language understanding systems are not advanced enough to extract meaning from a resource.

- It is more difficult to generate automatic identification of features and concepts from resources such as program codes, digital images and complex systems.
- Automatic indexing tends towards the simplistic approach and supporting discovery or even finding the required information will become increasingly non-manageable.

1.2 CINDI System

Building a good indexing system requires information extraction methods tailored to each specific environment. The semantics of the files in which the primary resource is stored will be exploited in order to extract and summarize the relevant information to support the resource discovery. To do this, the primary file type should be identified and then the extraction methods are applied to the file. A corresponding database should be created to maintain archives of Semantic Headers. This database could be searched by users to locate pertinent documents. The CINDI system (Concordia Indexing and Discovery system) is under development using this strategy.

The CINDI system provides a mechanism to register, search and manage the meta-information, with the help of easy-to-use graphical user interfaces. The CINDI system tries to avoid problems caused by differences in semantics and representation as well as incomplete and incorrect data cataloguing. It also tries to avoid the problems caused by the difference in index terms. The meta-data (Semantic Header) could be entered by the primary resource provider (contributors) who may either use the Automatic Semantic Header Generator (ASHG) to create a draft version for manual editing or who may

manually generate the semantic header. If the meta-data is entered by the contributors manually, the system offers a three-level hierarchy pull-down menu to let contributors choose the controlled terms for subjects. This meta-data is retrieved from the meta-data database (semantic header database). Alternatively, for some special format files (HTML, TEXT, LATEX, and RTF), the CINDI system can use ASHG, a software that generates some meta-information of the submitted document, ASHG would assist the user in the process by generating a draft version of the semantic header. As the contributor helps in this process by verifying and correcting the draft semantic header entry, there is the potential for its accuracy to be high.

1.3 Organization of this Report

This report presents the design and implementation of the graphical user interface (GUI) for the CINDI System based on the World Wide Web. Moreover, it emphasizes the GUI design and its implementation for the CINDI system.

Chapter 2 describes the general background of the CINDI system. It gives the reasons for the use of PHP, MySQL, and the Apache web server. Moreover, it describes this web-based application's architecture. It outlines the relationship among the web server, web browser, PHP, and MySQL database. Furthermore, it presents the system architecture, and the relationship among each sub-system. Chapter 3 gives the detail of the security sub-system. It shows how the system prevents illegal users from misusing the system.

Chapter 4 is the kernel of the whole system. It shows the design and architecture of the semantic header, and illustrates the resource registration procedure. Chapter 5 describes the search sub-system and the annotation sub-system.

A brief summary of the CINDI system, the contribution of this report, and the suggested future work is given in Chapter 6.

Chapter 2

Background

2.1 Graphical User Interface

2.1.1 GUI Design Principle

Graphical user interfaces have been used since the late 1960s. However, it did not start to become popular until 1981, when Xerox introduced the 8100Star workstation. The first truly commercial success of a GUI-based system was the Macintosh. GUIs were soon developed for UNIX workstations by Sun Micro systems and for PCs by Microsoft. By now, most computer users interact through GUI. GUIs are popular, because they make applications easier to learn and use. By standardizing the way users interact with a program and input information, errors are reduced. As a result, GUI applications not only are favored by end users, but also save companies both time and money by increasing productivity and ensuring accuracy [ZHOU 97].

Nowadays, almost everyone uses computers for their everyday activities. Highly interactive web-based applications are more and more popular. The implementations of these application interfaces represent more than half of the total code. They are becoming more complex, and are inserting themselves in more mission-critical roles. For the users

of these applications, ease-of-use has become a prime factor for the user interface design. Time is valuable. Users do not want to read manuals; they want to spend their time accomplishing their goals, not learning how to operate new systems. Usability is now also critical for the user interface design. User's demands on software have changed; they expect to be able to sit down and use software with little or no training. Thus, usability is a do or die decision for user interface design.

Users vary greatly in their computer experience, language, general literacy, memory, and vision. Since the CINDI system is a web-based application for general use, the interface of the CINDI system design uses the following guidelines.

Simplicity

The more options, advertisements, and distractions on each screen, the more likely a user is to get confused. So there are no advertisements and few graphics or images on the web site of the CINDI system.

Clear texts

Use clear, uncomplicated fonts. Do not make text too small. The same size font is used for the same kind of content. Also, the system try to keep the characters and images looking the same in different browsers and operating systems by avoiding the use of special characters and special formats.

Easily achieved user goals

Keep the number of steps to a minimum. The more mouse clicks, and the less keyboard input, the better. The CINDI system interface tries to minimize the steps that users have to follow to achieve their goals.

Preventing users from becoming lost

Provide landmarks and navigational cues to tell users where they are. The CINDI system provides the name of the subsections, where the users are. Moreover, the system seldom opens a new window when the user clicks a button or link. If users open too many new windows, they will easily get lost among those windows.

2.1.2 PHP

Since PHP has been employed for the implementation of the Graphical User Interfaces of the CINDI system, it's necessary to give a brief introduction of PHP.

2.1.2.1 Web application with PHP

For the Web application, returning pre-built documents can satisfy many client requests. However, a static result is not sufficient, and a page needs to be generated for each request. There are a number of reasons why web pages need to be built on-the-fly: [BROG 01]

- **The Web page is based on data submitted by the user.**

  For instance, the result page from search engines and order-confirmation pages at on-line stores are specific to particular user requests.

- **The Web page is derived from data that changes frequently.**

  For example, a weather report or news headlines page might build the page dynamically, perhaps returning a previously built page if it is still up to date.

- **The Web page uses information from corporate databases or other server-side sources.**

  For example, an e-commerce site could build a Web page that lists the current price and availability of each item that is for sale.

PHP can handle this built on-the-fly problem. PHP is a server-side scripting language designed specifically for the Web. Within an HTML page, you can embed the PHP code that will be executed each time the page is visited. The PHP code is interpreted based on the clients' request at the web server and generates a corresponding HTML or other output.

2.1.2.2 Strengths of PHP

Performance

PHP is very efficient. Using a single inexpensive server, you can serve millions of hits per day. [Well 01]

Database Integration

PHP has native connections available to many database systems. In addition to MySQL, you can directly connect to PostgreSQL, mSQL, Orcale, dbm, filePro, Informix, InterBase, and Sybase databases, among others. Using the Open Database Connectivity Standard (ODBC), you can connect to any database that provides an ODBC driver. This includes Microsoft products, and many others. [Well 01]

Built-in Libraries

Because PHP was designed for use on the Web, it has many built-in functions for performing many useful Web-related tasks. You can generate GIF images on-the-fly, connect to other network services, send email, work with cookies, and generate PDF documents, all with just a few lines of code. [Well 01]

Portability

PHP is available for many different operating systems. You can write PHP code on the free Unix-like operating systems such as Linux and FreeBSD, commercial Unix versions such as Solaris and IRIX, or on different versions of Microsoft Windows. [Well 01]

2.1.2.3 PHP & JSP, ASP, JavaScript, HTML

PHP has a number of advantages over many of its alternatives, for example, Java Servlets, Microsoft Active Server Pages (ASP), and JavaScript. Here are a few of them.

Active Server Pages (ASP)

ASP is a competing technology from Microsoft. The advantages of PHP are two-fold. First, the dynamic part is written in PHP, not VBScript or another ASP-specific language, so it is more powerful and better suited to complex applications that require reusable components. Second, PHP is portable to other operating systems and Web servers; you aren't locked into Windows NT/2000 and IIS. [BROG 01]

Java Servlets

PHP is more convenient to write (and to modify) regular HTML than to have a zillion println statements that generate the HTML. Plus, by separating the presentation from the content, you can put different people on different tasks: your Web page design experts can build the HTML using familiar tools and leave places for your PHP programmers to insert the dynamic content. [Well 01]

JavaScript

JavaScript is normally used to generate HTML dynamically on the client side, building parts of the Web page as the browser loads the document. This is a useful capability but only handles situations where the dynamic information is based on the client's environment. With the exception of cookies, the HTTP request data is not available to client-side JavaScript routines. And, since JavaScript lacks routines for network programming, JavaScript code on the client side cannot access server-side resources like databases, catalogs, pricing information, and the like. JavaScript can also be used on the server, most notably on Netscape servers and as scripting language for IIS. PHP is far more powerful, flexible, reliable, and portable. [BROG 01] [Well 01]

Static HTML

Regular HTML, of course, cannot contain dynamic information, so static HTML pages cannot be based upon user input or server-side data sources. PHP is so easy and convenient that it is quite reasonable to augment HTML pages that only benefit slightly by the insertion of dynamic data. [Well 01]

2.2 Information Retrieval

Information Retrieval (IR) is concerned with the representation, storage, organization and access of information. The first step in the retrieval process is for the user to state the information needed. This has to be done in a format that enables the IR system to understand it and to act on it. To facilitate the task of finding items of interest, libraries and information centers provide information users with a variety of auxiliary aids. Each incoming item is analyzed and appropriate descriptions are chosen to reflect the information content of the item. Retrieval effectiveness is typically measured by two metrics: precision, which is the percentage of the retrieved documents that are relevant to the information need, and recall, which is the percentage of relevant documents in the collection that is retrieved. [Fung 95]

2.2.1 Metadata

Metadata is the information, which records the characterization and relationship of the source data. It provides succinct information about the source data, which may not be, recorded in the source itself due to its nature or an oversight.

2.3 Networked Database

MySQL is a very fast, robust, relational database management system (RDBMS). A database enables the system to efficiently store, search, sort, and retrieve data. The MySQL server controls access to the data to ensure that multiple users can work with it concurrently, to provide fast access to it, and to ensure that only authorized users can obtain access. Hence, MySQL is a multi-user, multi-threaded server. It uses SQL (Structured Query Language), the standard database query language worldwide. MySQL has been publicly available since 1996. [DAVI 98] It has now won the Linux Journal Readers's Choice Award three years running. MySQL has many strengths, including high performance, low cost, ease of configuration and learning, portability, and the availability of the source code.

2.4 CINDI System Design

2.4.1 Web Application Architecture

Since the CINDI System is a web application, it is necessary to introduce the architecture of a web application. Usually, a web application contains a web browser. Users can type the address of the web site in a web browser. The web browser communicates with the web server through HTTP protocol. The web server returns a static web page to the web browser, or sends a request to the web base middleware. The web base middleware (PHP) processes the request, and makes a connection to the database. After the database processes the query request(s) from PHP, it returns the query results to it. The web base middleware formats these results in HTML form (web page), and returns it to the web server. The web server passes this web page to the web browser through HTTP protocol. We can see this procedure in Figure 2.4.1.

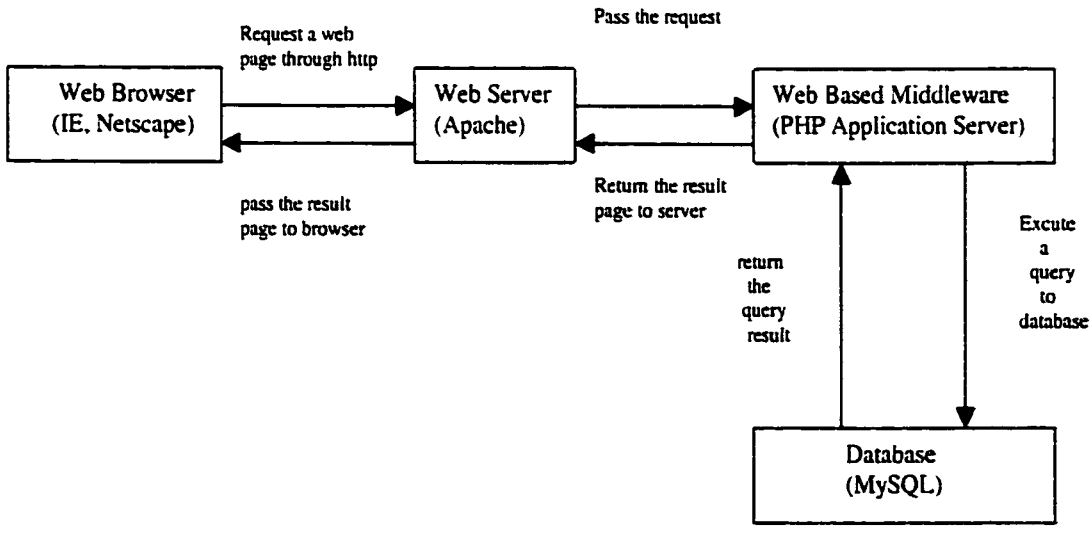

Figure 2.4.1 Web Application Architecture of the CINDI System

2.4.2 System Architecture

System design is the high-level strategy for solving the target problem and building a solution. System design includes decisions about the organization of the system into subsystems, the allocation of subsystems to hardware and software components, and major conceptual and policy decisions that form the framework for detailed design. The overall organization of a system is called the system architecture. [RBPE91]

System design is the first stage during which the basic approach to solving the problem is selected. System architecture can make a system design clearer. First of all, we should divide the system into sub-systems. A sub-system is not an object or a function, but a package of functions, associations, events, and constraints that are interrelated and that have a reasonably well-defined and small interface with other sub-systems. [RBPE91]

The CINDI system is divided into six sub-systems: the security sub-system (SC), the resources registration sub-system (RR), the search sub-system (SE), the database sub-system (DB), the semantic header auto generator sub-system (ASHG), and the annotation sub-system (AT). Figure 2.4.2 shows this architecture of the CINDI system.

System contributors and users register their personal information using the security sub-system, and get permission to use the system. Contributors use the resources registration sub-system to register their resources. Users use the search sub-system to search for useful information, and use the annotation sub-system to post their comments. If the

resource file is in text, html, latex, or rtf format, the semantic header auto generator sub-system will generate a draft semantic header. Through the RR sub-system, it allows the contributors to accept the draft or modify it before registering it.

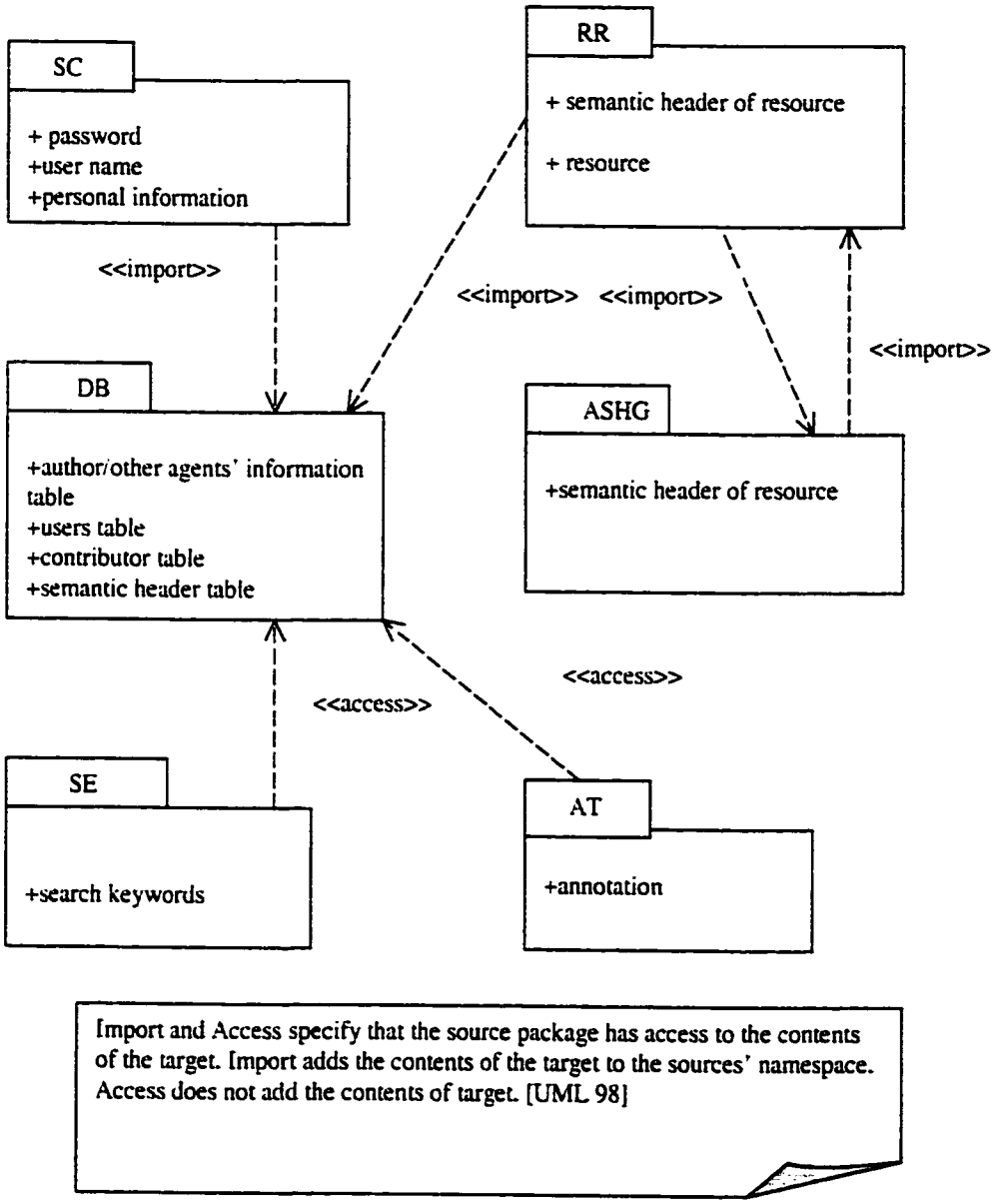

Figure 2.4.2 UML Model of Architecture of the CINDI System

Chapter 3

Security Sub-system

3.1 Security of the System

When considering security, the first thing that needs to be evaluated is the importance of what the system is protecting. The importance of information to the system owners and potential crackers should both be considered. The value of information stored on the computer of a casual user, a business, a bank, and a military organization obviously varies.

It might be tempting to believe that the highest possible level of security is required for all sites at all times, but protection comes at a cost. Before deciding how much effort or expense security warrants, the value of the system information should be considered. Then the security system is built.

Most casual users will probably have limited time to learn about or work towards securing their systems. Given that information stored on their machines is likely to be of limited value to anyone other than its owner, attacks are likely to be infrequent and involve limited effort. However, all network computer users should take sensible

precautions. Even the computer with the least interesting data still has significant appeal as an anonymous launching pad for attacks on other systems.

The CINDI system is an academic, non-profit system. It does not deal with sensitive information. So its security level need not be high. However, the information has value and must not be modified or deleted by malicious visitors. Moreover, it needs to prevent malicious visitors from posting unsuitable information on the system. So the system needs to provide some certainty about who the system is dealing with, and make sure the information the system receives is from a trusted primary resource provider. For all these reasons, there is a need for a security system.

Authentication is one way to prove that somebody is actually who he claims to be. There are many possible ways to provide authentication-for example, passwords, digital signatures, biometrics measures such as fingerprint scans, and measures involving hardware such as smart cards. However, only two are in common use on the web: passwords and digital signatures. As we know, with many security measures, the security systems are more troublesome to use. Passwords are simple to implement, and easy to use. They require no special input devices. Even though it is quite simple, it fits the CINDI system's requirement. Moreover, passwords are sent to users to their email addresses. This ensures that users' email addresses really exist. Furthermore, we implement authentication to the system with Session Control (the detailed description is given in section 3.3). Figure 3.1 shows the first screen of the CINDI system. It splits the system users into two groups: contributors and users.

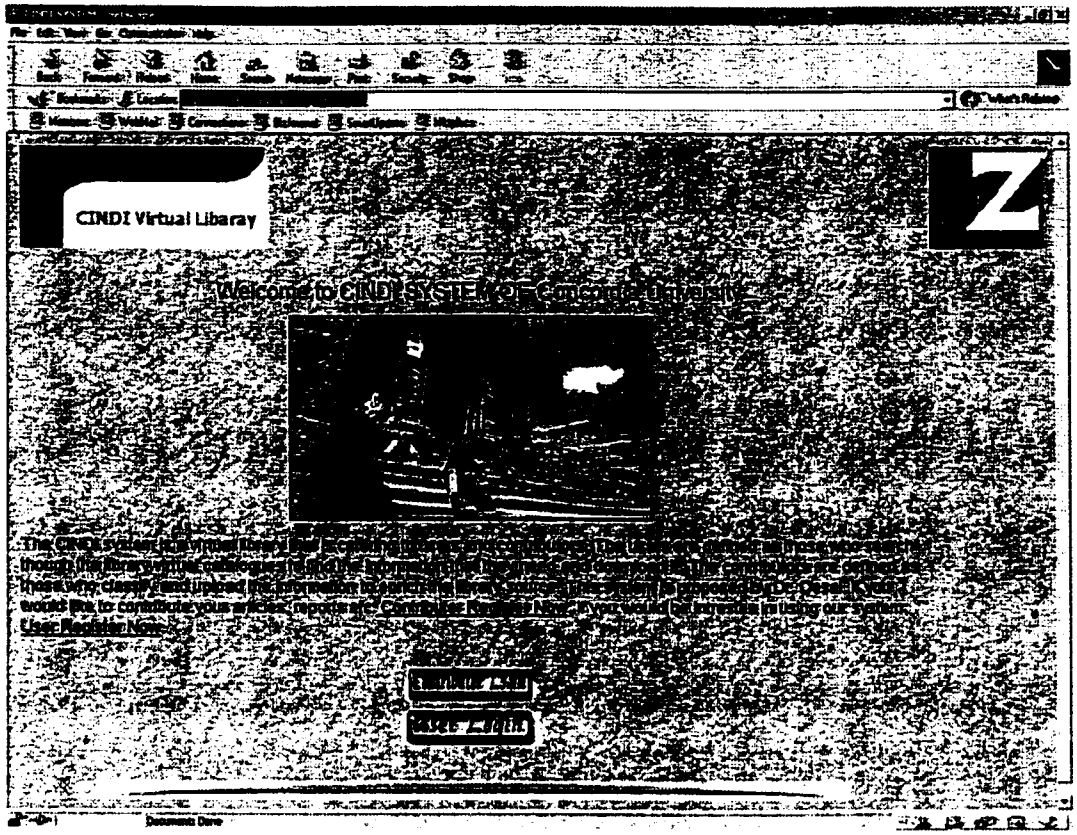

Figure 3.1 Home Page of the CINDI System

3.2 Registration

The security sub-system contains two parts: The contributor registration sub-system and the user registration sub-system. The contributor registration sub-system lets the primary resource providers register their personal information in the system. Thereby, the providers are registered as contributors of the system. After they input their personal information and pick a user ID (the user ID should be unique), the system will send them an email to confirm their user name and password. After they receive the email, they can login into the system and upload the documents. Even though the PHP script can check if the format of the email address is correct or not, it cannot prevent a malicious user from inputting a correct format but non-existent email. Confirmed email can avoid this problem very well. If the users input a fake email address, they cannot receive the confirmation email. As a result, they cannot get the password to login into the system. Of course, this scheme doesn't give the system a lot of information about the user other than the users' email addresses; but it at least gives the system a clue to trace the malicious users.

Figure 3.2 shows the login user interface. If users have already registered, they can simply input their user name and password to login in to the system. Alternatively, they can click the "register now" button to register. This user interface is almost the same for the contributor registration sub-system and the user registration sub-system.

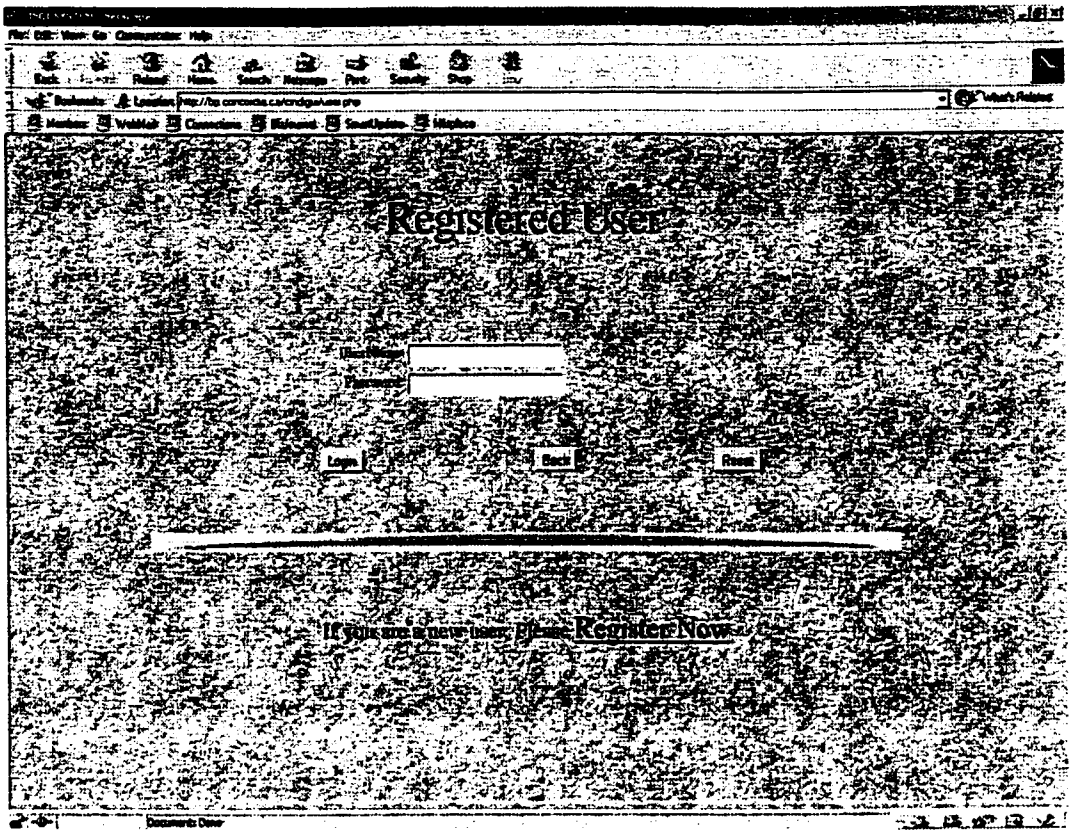

Figure 3.2 Login-in User Interface of the User Registration Sub-System

3.2.1 Contributor Registration Sub-System

Figure 3.2.1 shows the user interface of this contributor registration sub-system. We can see that the username, first name, last name, and email address are essential input fields: The system will remind the contributors if they do not fill in these fields. Also, the system will check the database if the username that the contributors choose is already in the database. If the user name already exists in the database, the system will suggest that the contributors pick another username. After the contributors submit their personal

information, the system will generate an eight-character password, which includes number, capital letter, and lower-case letter. Subsequently, the system sends a confirmed email containing the username and the password to the contributor.

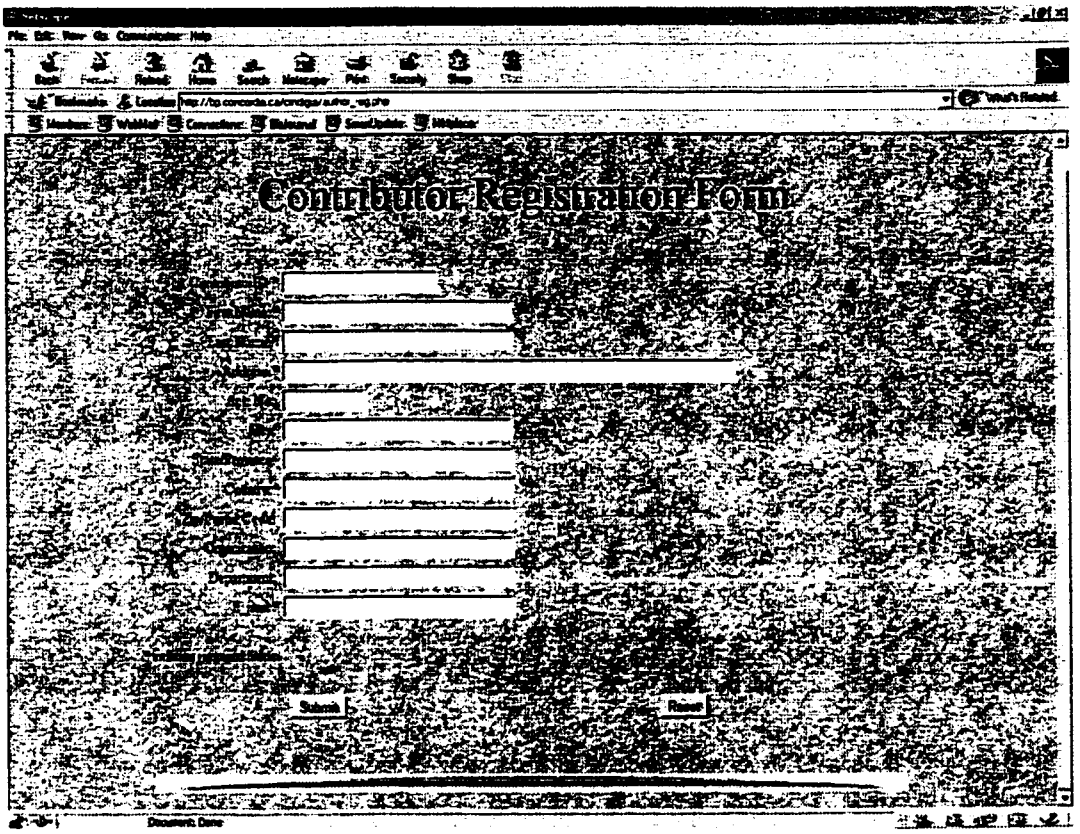

Figure 3.2.1 User Interface of the Contributor Registration Sub-system

3.2.2 User Registration Sub-System

The user registration sub-system lets the system users provide their personal information to the system, and register as users of the system. After they input their personal information and pick the user name, the system sends them an email to confirm their user

name and password. After they receive the email, they can login into the system and search the information they need. Moreover, they can write their annotation for resources in the CINDI virtual library.

The security sub-system of CINDI system collects the users' personal information for statistical reasons, rather than for security purposes. The CINDI system tries to use this information to collect statistics about the users of this virtual library. This survey functionality is not implemented in the current system. This is a part of the future work of the CINDI system. The user interface of user registration sub-system is almost the same as the contributor registration sub-system. Figure 3.2.2 shows the interface of users registration. The difference between these two sub-systems is that the user registration sub-system does not have as many required fields as the contributor registration sub-system.

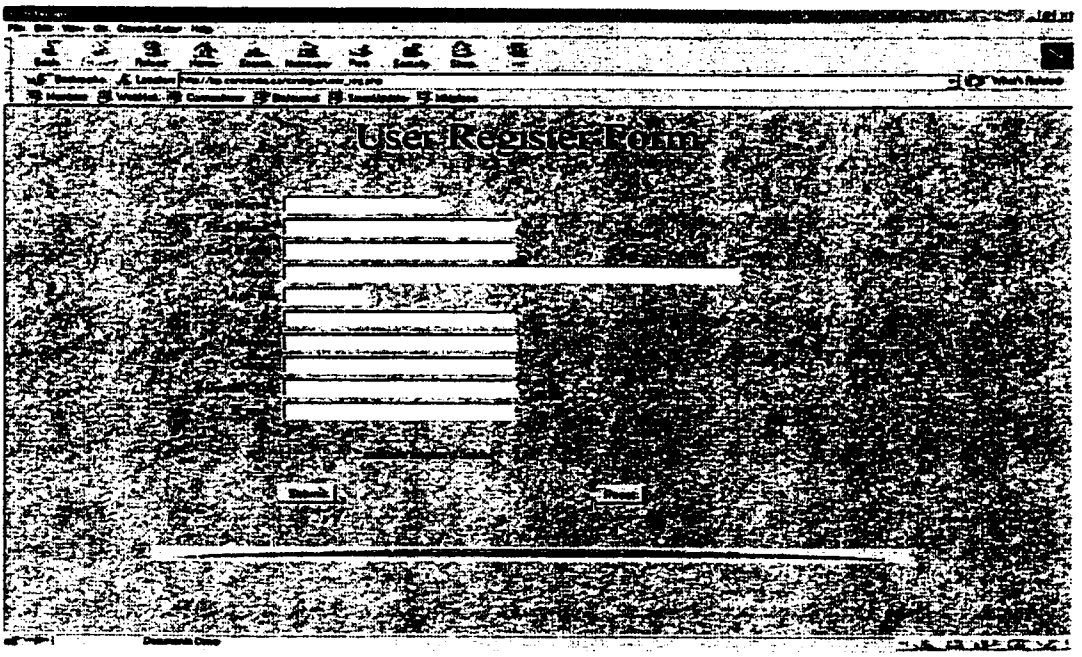

Figure 3.2.2 User Interface of the User Registration Sub-

3.2.3 Error Handling during Registration

Human beings will make mistakes. Of course, the system users will make mistakes too. There are several kinds of mistakes caused by users: miss-input, duplicate information, wrong format etc. The interface can handle these errors in the following ways. Since the user ID should be unique for the system, the users/contributors registration sub-system will check the corresponding databases and figure out if the user ID is already assigned. If the user ID that the users/contributors pick is already in the database, the interface will show the error message: "User ID already exists. Please choose another one". If the users/contributors do not input the required fields, the system will notify them by the corresponding error message. Figure 3.2.3 shows these error messages. Moreover, the PHP script checks the format of inputs. For example, the email address should have a correct format. If users/contributors input a wrong email address format, the system will also notify the users/contributors. Furthermore, the system provides another chance for the users/contributors to register again. Of course, the users/contributors can exit the system without storing any information in database if they don't want to provide the required personal information.

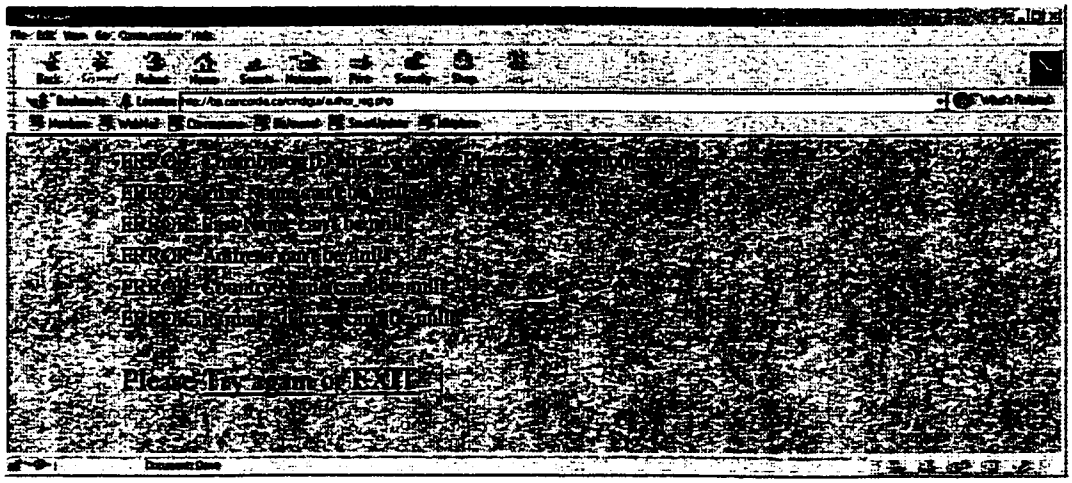

Figure 3.2.3 Error Handling of the Registration Sub-system

3.3 Session Control

A malicious visitor can just type the URL address of the page he wants in the browser, and avoid the authentication system. It means if the users know what page they want to go to, for example "register.php", they can just type this address in the browser and go to this page. The register security sub-system will be rendered useless. The CINDI system use session control to avoid this problem. Each visitor accessing the CINDI system is assigned an unique id (we use user ID in the CINDI system) after they login. This is stored in a cookie on the client side. The system will check the user ID at the beginning of each page. If the page does not detect the user ID, it will not show the correct page content to users. Instead, the page will show a warning, and ask the user to login. Figure 3.3.1 shows the correct page if the contributors login correctly. Figure 3.3.2 shows the following scenario: If a malicious visitor types "bp.concordia.ca/cindigui/register.php" in the browser directly, the correct page cannot be seen. Instead, a warning message appears.

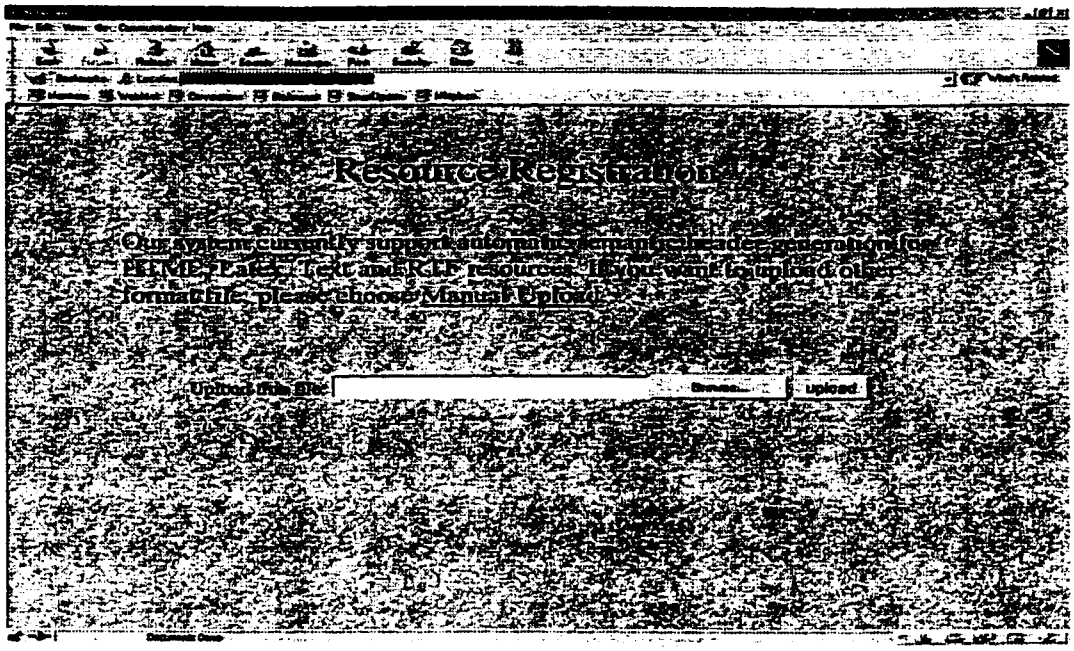

Figure 3.3.1 Correct Page for Legal Users

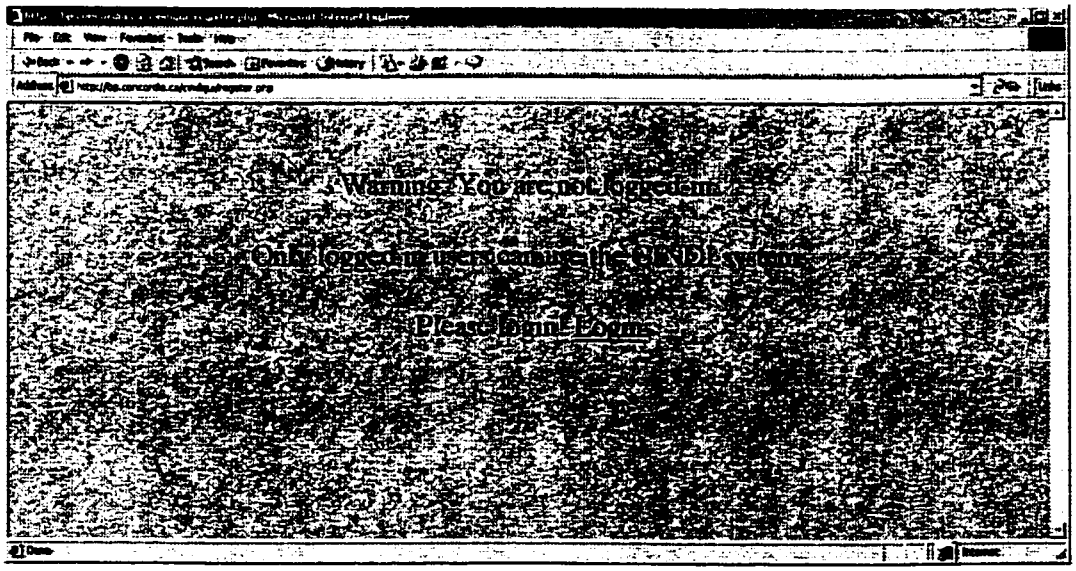

Figure 3.3.2 Warning Page for Malicious Visitor

Chapter 4

Resource Registration Sub-system

The resource registration sub-system offers the users/contributors a user-friendly graphical interface to register the resources. For the contributors who want to upload HTML, TEXT, RTF, and LATEX format files, the resource registration sub-system uses automatic semantic header generator (ASHG) to help them get the semantic header of the resources. Alternatively, for other format files, contributors must enter the semantic header of the resource by themselves. For this reason, we separate the resource registration sub-system into two parts: manual registration, and auto registration. Figure 4.1 shows the interface of the resource registration sub-system.

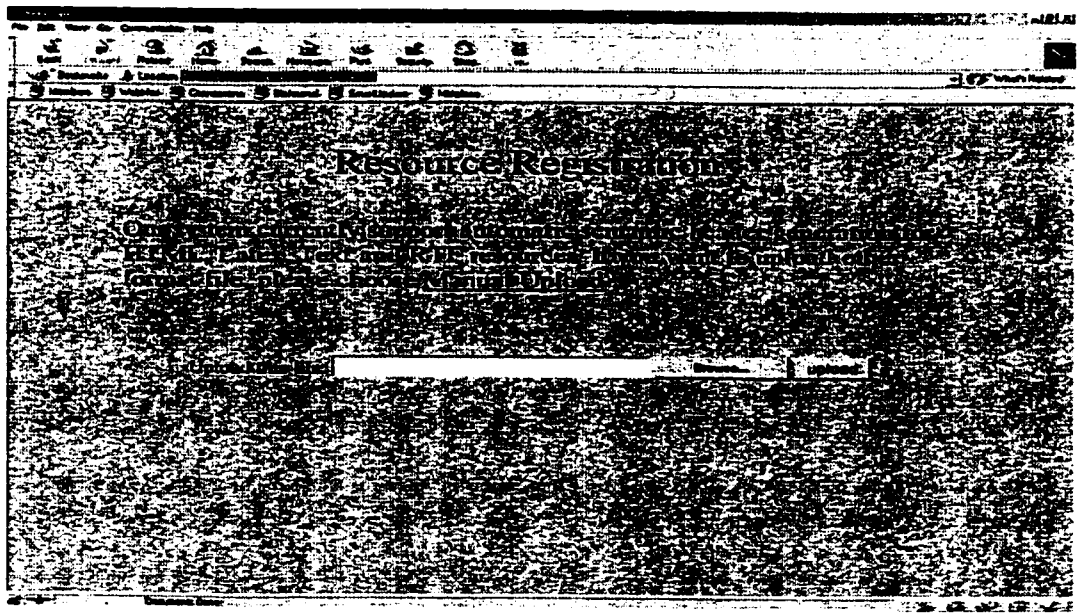

Figure 4.1 Interface of the Resource Registration Sub-system

4.1 Semantic Header

As mentioned before, cataloguing information is very important for a virtual library. The efficiency and effectiveness of a virtual library almost always depends on good indexing and cataloguing of the resources. For good cataloguing and indexing, the CINDI system uses a meta-data description for resources called the Semantic Header proposed by Dr Desai [BCD 95]. The Semantic Header includes those elements that are most often used in the search for an information resource. Since the majority of searches begin with a title, name of authors (70%), subject and sub-subject (50%) [KATZ], the CINDI system requires the entry for these elements in the Semantic Header. Similarly, the abstract and annotations are relevant in deciding whether or not a resource is useful. Therefore, they are also included. [SHAY 97] [BCD 95]. Figure 4.1.1 shows the whole design for the semantic header.

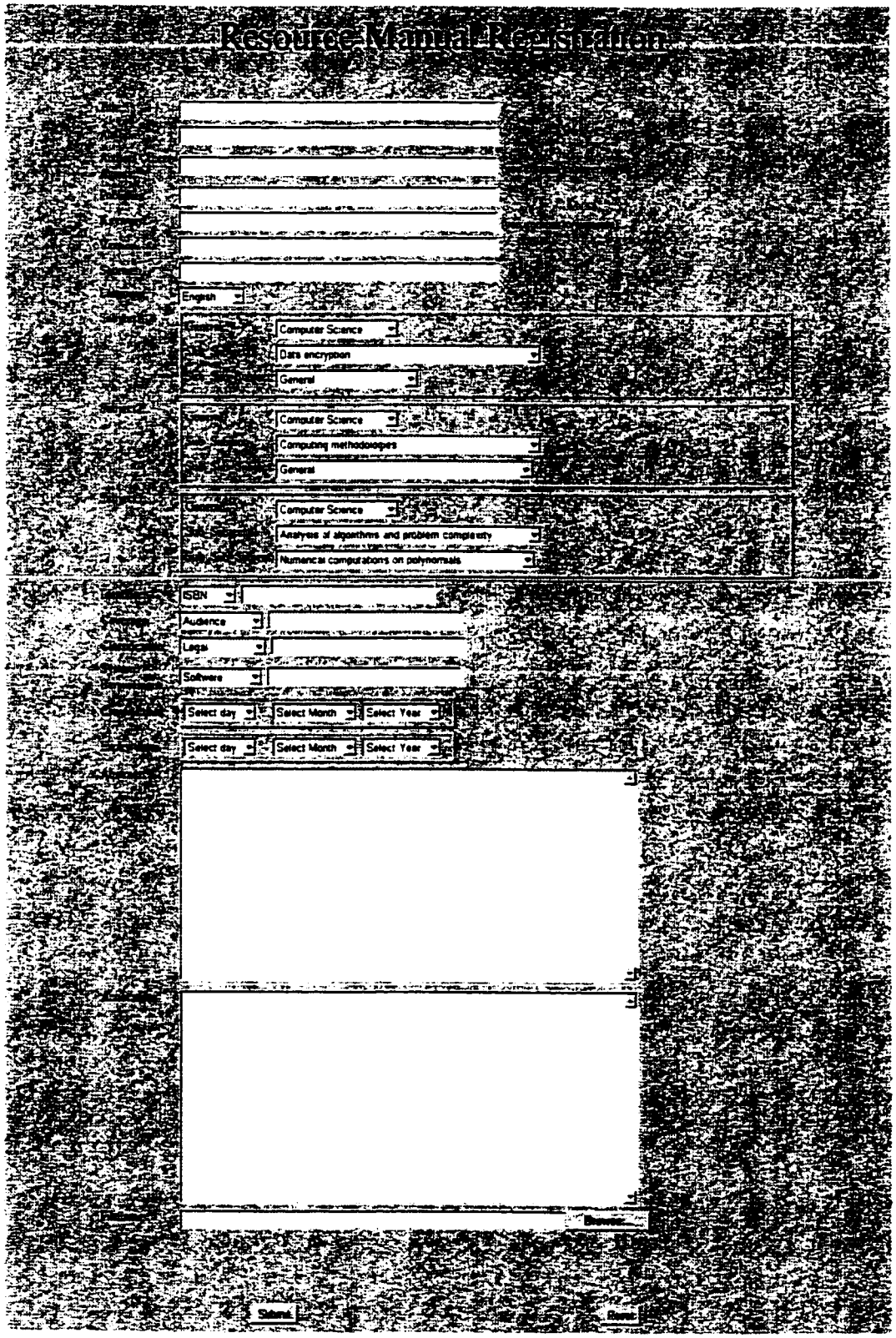

Figure 4.1.1 Semantic Header Interface for Manual Registration

4.1.1 Semantic Header Structure

The structure of the semantic header using extended BNF rules [GEHA 86] is shown below. Here, tags are given in quotes, optional items are bracketed by square brackets, alternative items are separated by a bar "|", and the superscript plus sign "+" means that items may be repeated one or more times.

semantic_header	:= "<semantichdr>" contents " </semantichdr>"
contents	:=title al-title authors publisher keywords version source language subjects identifier coverage classification system_requirement genre dates abstract annotation
title	:= "<title>" ..."</title>"
alt-title	:= "<alttitle>"[...]"</alttitle>"
authors	:= "<author>"author_info</author>"
author_info	:= role name organization address phone email
role	:="<role>" Author|Co_author|Editor|Artist|Designer| Programmer"</role>"
name	:= "<name>"[...]"</name>"
organization	:= "<organization>"[...]"</organization>"
address	:= "<address>"[...]"</address>"
phone	:= "<phone>"[...]"</phone>"
email	:= "<email>"[...]"</email>"
keywords	:= "<keywords>"[...]"</keywords>"
version	:= "<version>"[...]"</version>"

source := "<source>"[...]"</source>"

language := "<language>"[Arabic|Chinese|English|French|
German...]"</language>"

subjects := "<subject>" subject_group+ "</subject>"

subject_group := general sub_subject sub_sub_subject

general := "<general>" ... "</general>"

sub_subject := "<sub_subject>"[...]"</sub_subject>"

sub_sub_subject := "<sub_sub_subject>"[...]"</sub_sub_subject>"

coverage := "<coverage>"coverage_group"</coverage>"

coverage_group :=coverage_type coverage_value

coverage_type := "<coverage_type>"Geographical|Spartial|Temporal|
Epoch"</coverage_type>"

coverage_value := "<coverage_value>"[...]"</coverage_value>"

identifier := "<identifier>"identifier_group"</identifier>"

identifier_group :=identifier_type identifier_value

identifier_type := "<identifier_type>"FTP|ISBN|ISSN|Gopher|
HTTP|URN|SHN|Call No."</identifier_type>"

Identifier_value := "<identifier_value>"[...]"</identifier_value>"

classification := "<classification>"classification_group"</classification>"

classification_group :=classification_type classification_value

classification_type :="<classification_type>"Legal|
Security Level"</classification_type>"

classification_value := "<classification_value>"[...]"</classification_value>"

system_req := "<system_req>"system_req_group"</system_req>"

system_req_group :=system_req_type system_req_value

system_req_type := "<system_req_type>"NetworklSoftwarel

Hardware."</system_requ_type>"

System_req_value := "<system_req_value>"[...]"</system_req_value>"

genre := "<genre>"file_format file_size"</genre>"

file_format := "<file_format> ... "</file_format>"

file_size := "<file_size>" ... "</file_size>"

dates := "<dates>"create_date expiry_date upload_date"</dates>"

create_date := "<create_date>"[...]"</create_date>"

expiry_date := "<expiry_date>"[...]"</expiry_date>"

upload_date := "<upload_date>" ... "</upload_date>"

abstract := "<abstract>" ... "</abstract>"

annotation := "<annotation>" [...] "</annotation>"

4.1.2 Semantic Header Detail Description

For each of these fields, the description is given as follow:

- **title**

Title field contains the title of the paper, which is given by the creator. It is a required field. Contributors must provide this field. If they do not fill in this field, the system will notify the contributor by an error message (figure 4.1.2).

- **alt-title**

The alternate title (alt-title) field is used to indicate a secondary title of the resource. It is an optional field. Contributors can decide whether to provide the information of this field or not.

- **authors**

Authors (other agents) field provides the contributors a place to input the authors or other agents' name of the resource. It is a required field. If it is no vale been provided, the system will notify the contributor by an error message (figure 4.1.2). The interface allows users to input the role and the personal information of the authors or other agents too. This procedure is described in detail in section 4.1.3.

- **publisher**

The publisher field provides a place that allows the contributor to input the publisher of the resource. Since not all resources have been published, it is an optional field.

- **keyword**

The keyword field can contain a list of keywords mentioned in the resources, or provided by the contributors. It is an optional field, since the contributors may not know the keywords of the resources or some resources may not have keywords at all.

- **version**

This field contains the version of the resource. It is also an optional field.

- **language**

This field let the contributors choose the language that the resources use. The system provides a pull-down menu that contains all the possible language names. This solution

can prevent the contributors from miss-inputting the language name. It is good for index standardization.

- **subject**

The subject field contains three groups of three-level hierarchy subject pull-down menu. Each subject group contains general, sub-subject, and sub-sub-subject fields, thereby allocating resources to the appropriate subject domain. These three groups of three-level hierarchy subject pull-down menu are based on an expert standard. For the Computer Science subject hierarchy, the system uses the ACM indexing standard. For the Electrical Engineering subject hierarchy, the system uses INSPEC standard.

- **identifier**

This field contains two sub-fields: identifier type and identifier value. It lets contributors input the identifier of resources, for example the ISBN number, etc.

- **classification**

This field contains two sub-fields: classification type and classification value. It indicates the security or other restriction of the resources.

- **coverage**

This field contains two sub-fields: coverage type and coverage value. Contributors can input the coverage of the resources.

- **system_req**

This field contains the system requirement for the resources. It has two sub-fields: the system requirement type and the system requirement value. Contributors can input the system requirement for the resources in this field.

- **dates**

The date field contains three sub-fields: create date, expiry date, and upload date. The interface just asks the contributors to input create date and expiry date of the resource, but not the upload date. The system will auto generate the upload date base on the real date of the resource upload, and store it in the semantic header database.

- **genre**

This field contains the two sub-fields: file format and file size. The user Interface does not ask the contributor to input these two sub-fields, but auto generates them by the system. This can prevent the contributor miss-input the format and size, and increase the precision of the semantic header.

- **abstract**

This field contains the abstract of the resources. It is a required field. If it is null, the system will notify the contributor by an error message (figure 4.1.2).

- **annotation**

This field contains the annotation of the resource. It is an optional field.

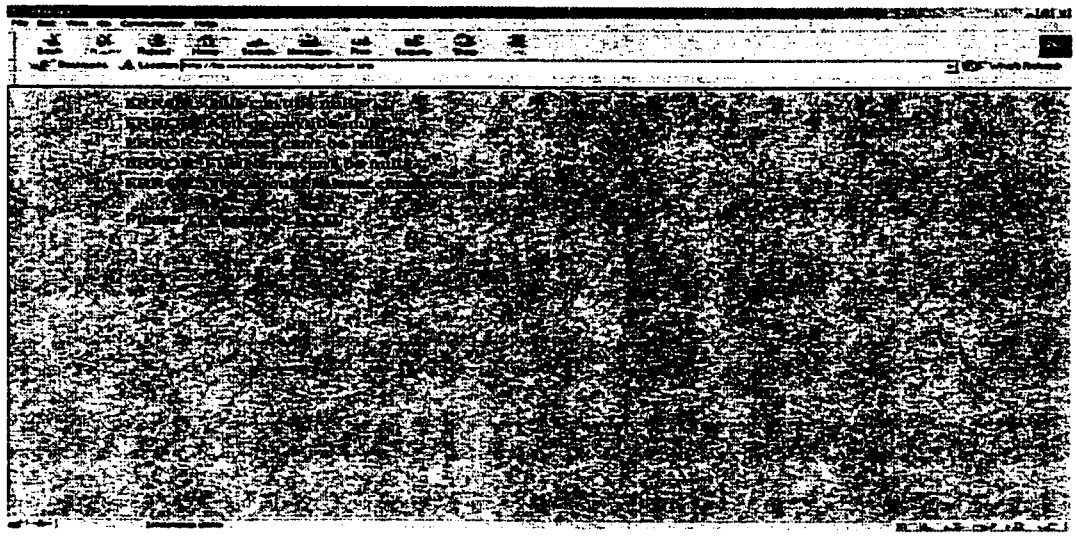

Figure 4.1.2 Error messages for the Required Fields

4.1.3 User Interface of Authors/Other Agents Field

After contributors submit their resources, the system saves these resources, and stores the location of these resources with the semantic header information into the semantic header table. After this, the system will depend on the input of the authors (other agents) field to ask the details of authors or other agents. Figure 4.1.3.1 shows this interface.

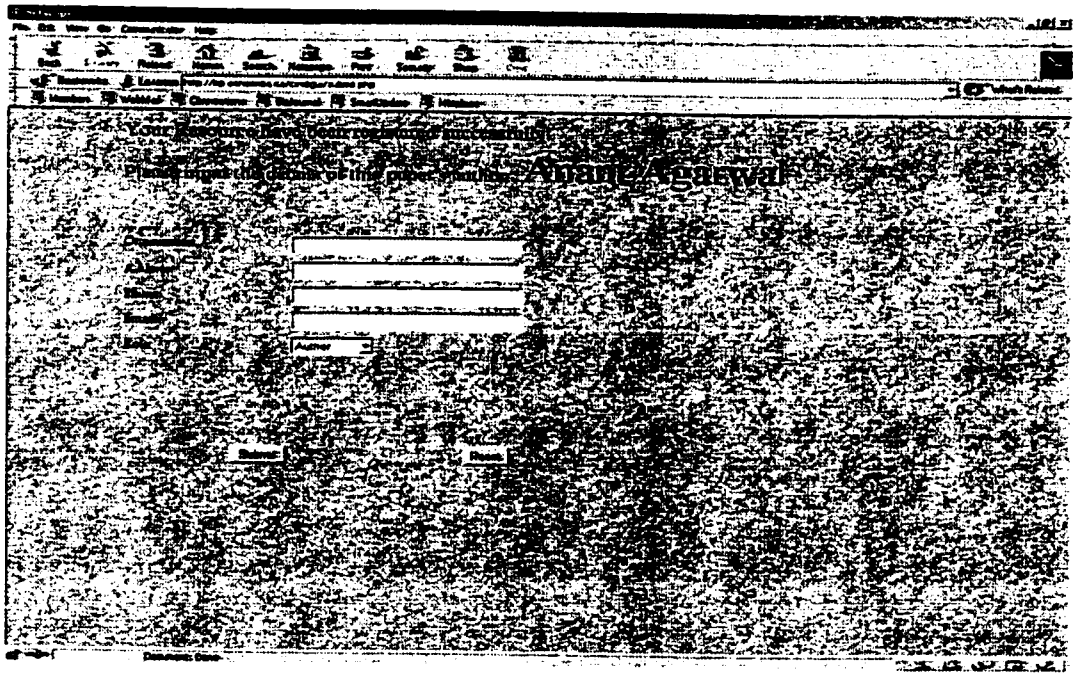

Figure 4.1.3.1 Interface of Personal Information of Authors/Other Agents: page1

Contributors can input the personal information: organization, address, phone, email address, and the role of this person. All these fields except the role field are optional since the contributors may not know the personal information of this person. The system provides a pull-down menu for the role field. This prevents the contributors from inputting a wrong value, and is good for the standardizing of the indexing. After

contributors enter the personal information of this person, the system will go to the database and check if this person is in the database. Since different persons can have same name, the system cannot use only the name of the person to identify the record is same or not. Moreover, a person may belong to a different company at different periods. For example, an individual could have worked for IBM ten years ago, but currently he is working for HP. We cannot just depend on one of the detail fields to identify if the personal record is the same. The system must use all the personal information fields to identify if it is a new record or not. If the person is not in the database, the system adds this person to its database. If the person is already in the database, the system will tell the contributor that the person is already in its database and does not add it on. User interface will prompt for the personal information of all authors or other agents of the resource. Figure 4.1.3.2 & Figure 4.1.3.3 show this procedure.

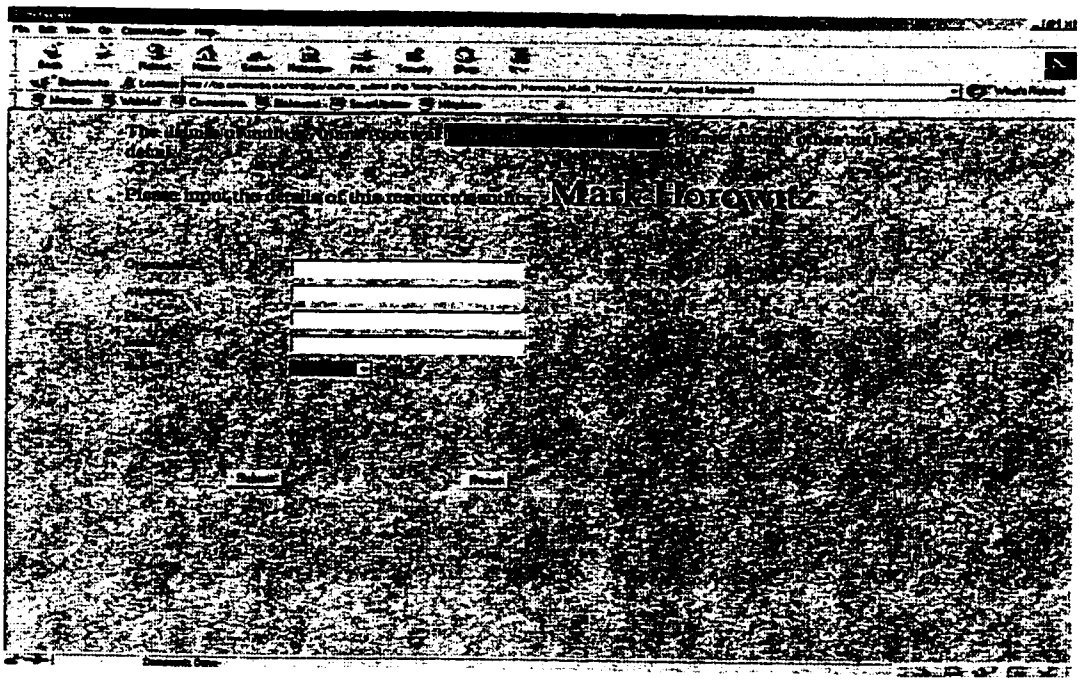

Figure 4.1.3.2 Interface of Personal Information of Authors/Other Agents: page2

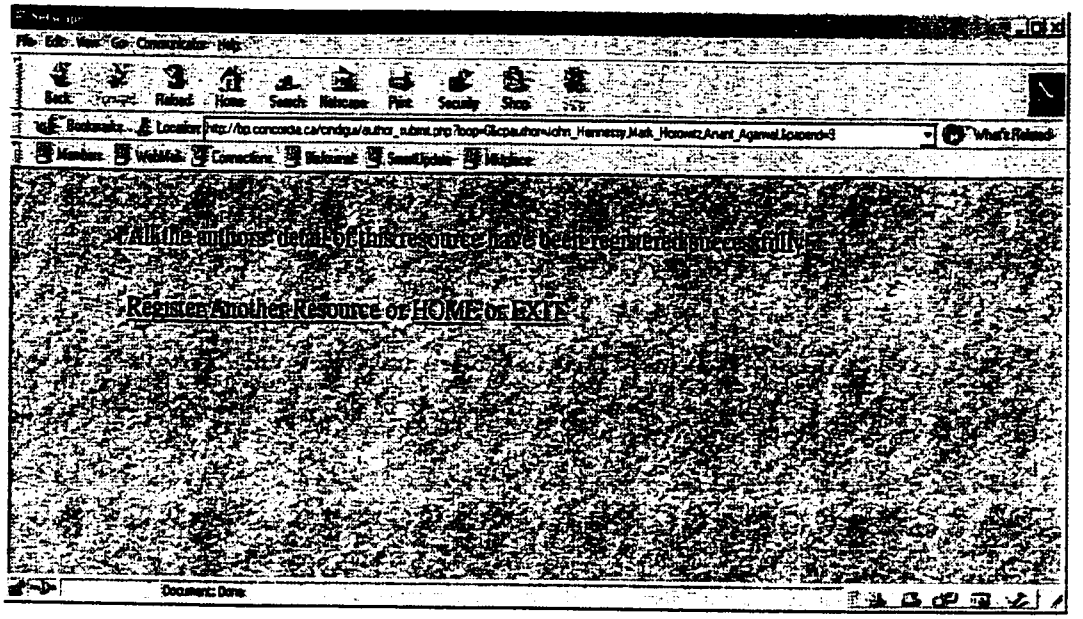

Figure 4.1.3.3 Interface of Personal Information of Authors/Other Agents: page3

4.2 User Interface for Resources Registration

As figure 4.1 shows, the resources registration sub-system is split into manual registration and auto registration. After the contributors click the manual upload link, the system will go to the manual upload sub-system as Figure 4.1.1 shows. Alternatively, if the contributors upload the HTML, LATEX, TEXT and RTF format files, the system run the ASHG to generate a draft semantic header, and shows it to the contributors. Contributors can modify it, and submit it to the database. This can facilitate the registering process. The user interface of the auto registration is shown in figure 4.2.1.

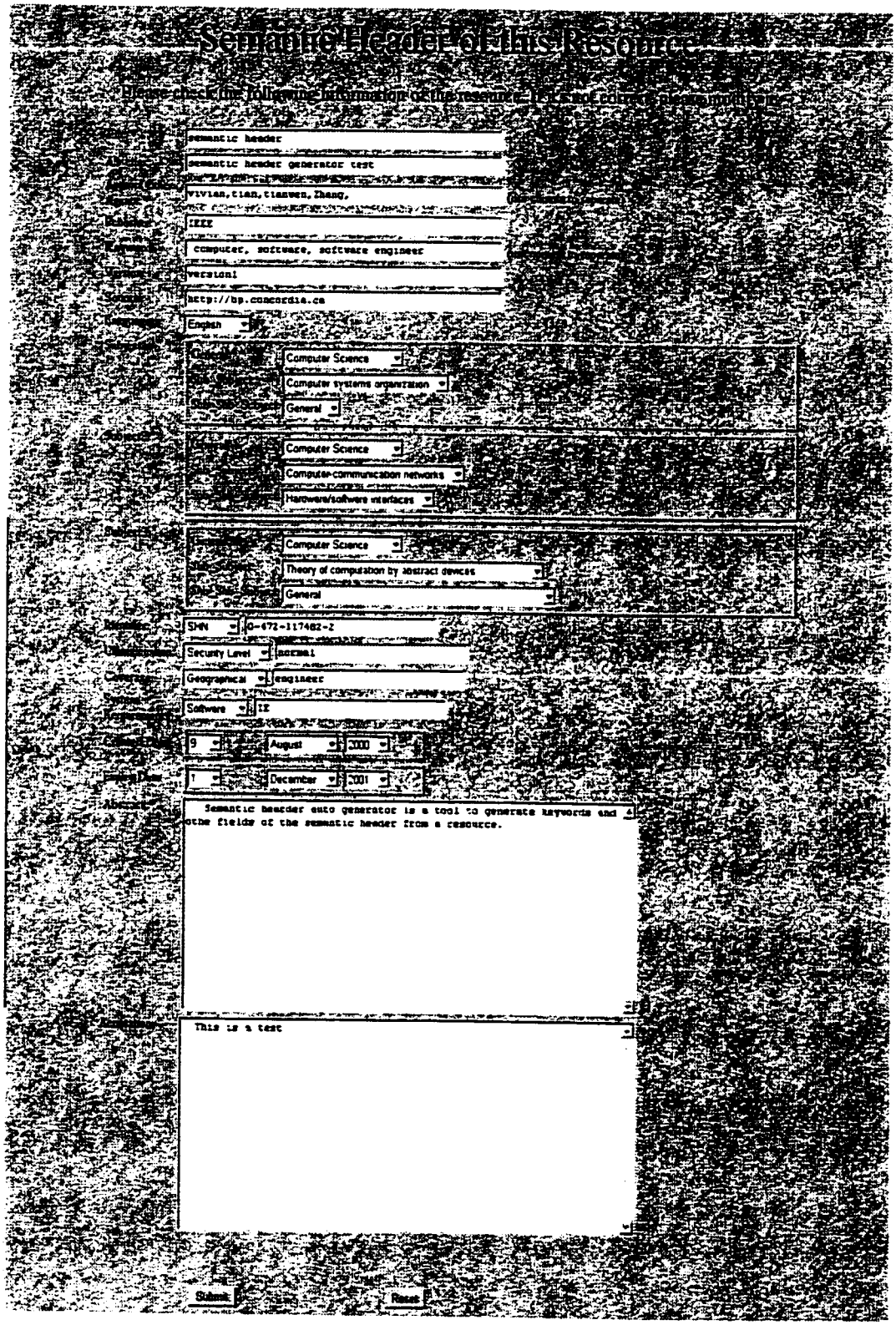

Figure 4.2.1 User Interface of Auto Registration

The personal information of authors or other agents is auto-generated by ASHG too, and contributors can modify them also. The procedure is almost the same as the manual paper register.

4.3 Using Pull-down Menu in User Interface

The behavior of humans is most unexpected. We do not know what the users of the system will input in a field. With an increasing number of user input, there is a higher possibility of errors. The CINDI system uses two ways to try to solve this problem: automatic semantic header generator (ASHG) and the pull-down menu. ASHG can just handle four kinds of files currently. So in this interface we use as many pull-down menus as possible. A pull-down menu can prevent incorrect inputs and provide standardized terms. Also, the menu selection can shorten the training time for novices, reduce keystrokes, provide rapid, accurate entries of items from standard choices. The significant and the biggest contribution of this report is the three groups of the three-levels hierarchy subject pull-down menu. Once the contributors choose the first level subject, the second level subject pull-down menu can show the corresponding sub-subject of the subject that the contributor chose. After the contributors choose the second level subject, the third level subject pull-down menu shows the corresponding sub-sub-subject of the second level subject. For example, if the contributor chooses "Computer Science" as the first level subject, the second level pull-down menu will just show "Software ", " Information storage and retrieval ", " Software engineering". The second level subjects belonging to other first level subjects will not be shown. After the second level subject has been chosen (for example, "Information storage and retrieval"), the third level subject pull-

down menu will show the sub-sub-subjects belonging to this sub-subject (" Information search and retrieval", "Retrieval models in information search and retrieval", " Query formulation in information search and retrieval"). This really helps the standardization of a bibliography, and prevents the wrong catalog. Offering contributors a standard catalog and a correct hierarchy of subjects, sub-subjects, and sub-sub-subjects to choose from increases the search speed, and prevents mistake. Since contributors may not know which document belongs to which subject, sub-subject, and sub-sub-subject, they may make a mistake if there is no hierarchy pull-down menu.

Chapter 5

Search/Annotation Sub-system

5.1 Resource Search

The CINDI system also offers a sub-system to let users search for useful information for them. Users can use the title of the resource, the author or authors of the resource, keywords of the abstract or keywords of the resource, and the subject to search the resources they want. The query can be implicit or explicit. An explicit query means the user knows the exact title, authors, keywords, and subject of the resource that he is searching. The implicit query means the user just has vague information about the resource; for example, he may just know what subject the resource belongs to or part of the author's name, title, or some keywords that may appear in the resource.

5.1.1 User Interface for Search Sub-system

The graphical user interface of this search sub-system is shown in Figure 5.1.1.1. Users can input the title of the resource, the authors of the resource, the keyword of the resource, and the subject, sub-subject, and sub-subject of the resource. Alternatively, users can just input only the fields that they know. The system can search for all the

semantic headers that fit these requirements. The system uses the three-level hierarchy pull-down menu for controlled subject terms. This can prevent the input of arbitrary subject leading to incorrect results.

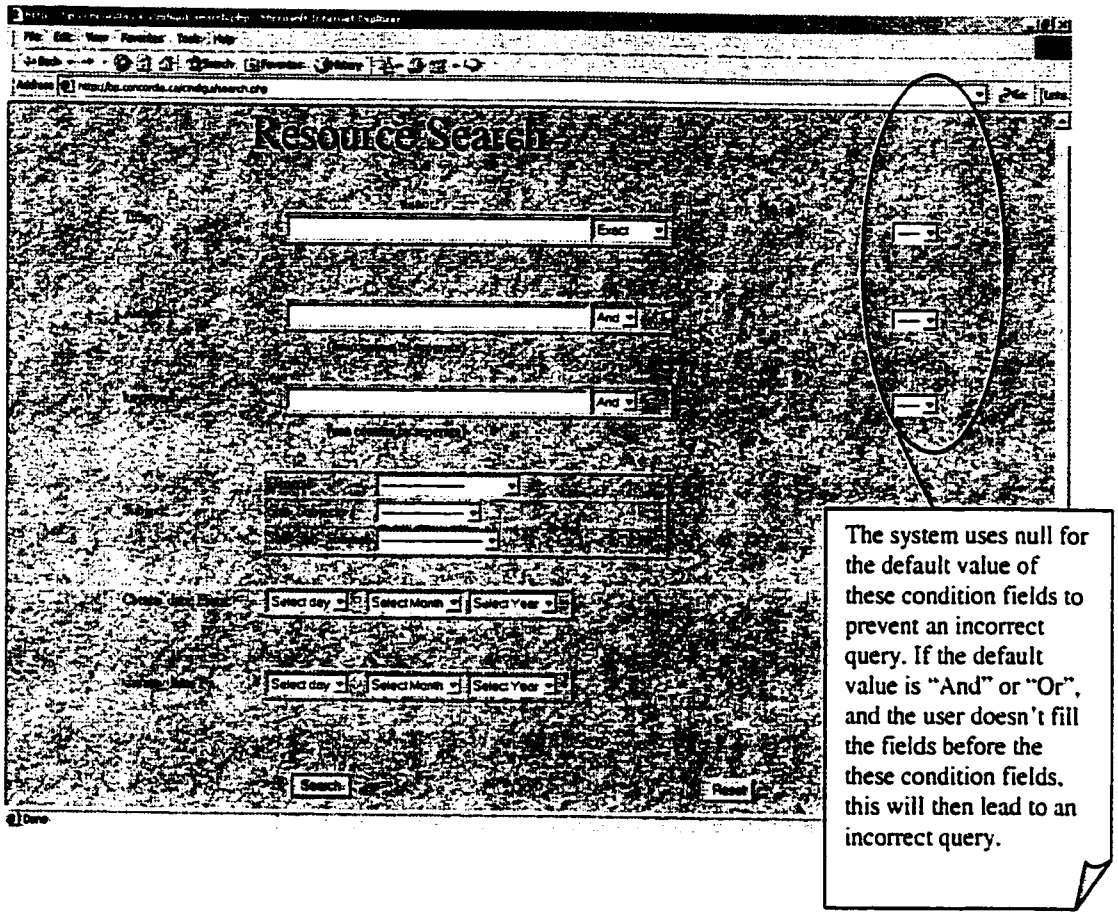

Figure 5.1.1.1 User Interface of Resource Search Sub-system

After users click the "Search" button, the system searches the corresponding information in the database. If the system cannot find the information that the users want, the system will show a page telling the resource they want cannot be found. Figure 5.1.1.2 shows the GUI of this page.

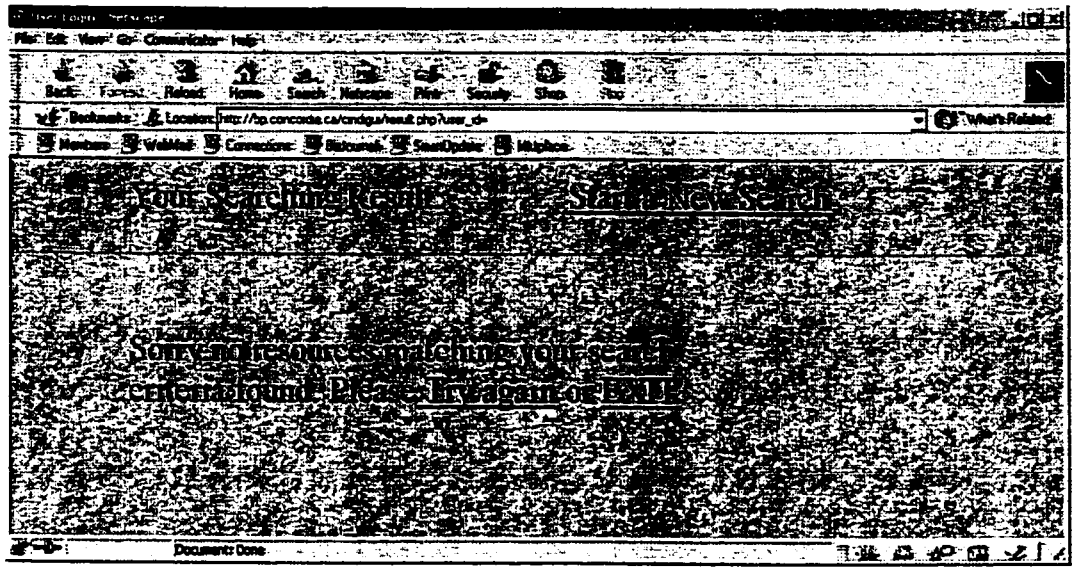

Figure 5.1.1.2 User Interface of Search Result (No match)

If the system can find the resources that users want, it returns a page to show the resources' information, such as title, author(s), subject, sub-subject, sub-sub-subject, and abstract of the documents. Moreover, the system can let the users either download the resources they want or open the resources in the browser. To achieve this goal, the users just need to click the link in the "document" field. Figure 5.1.1.3 shows this interface. Figure 5.1.1.4 shows the interface after the users click the document field.

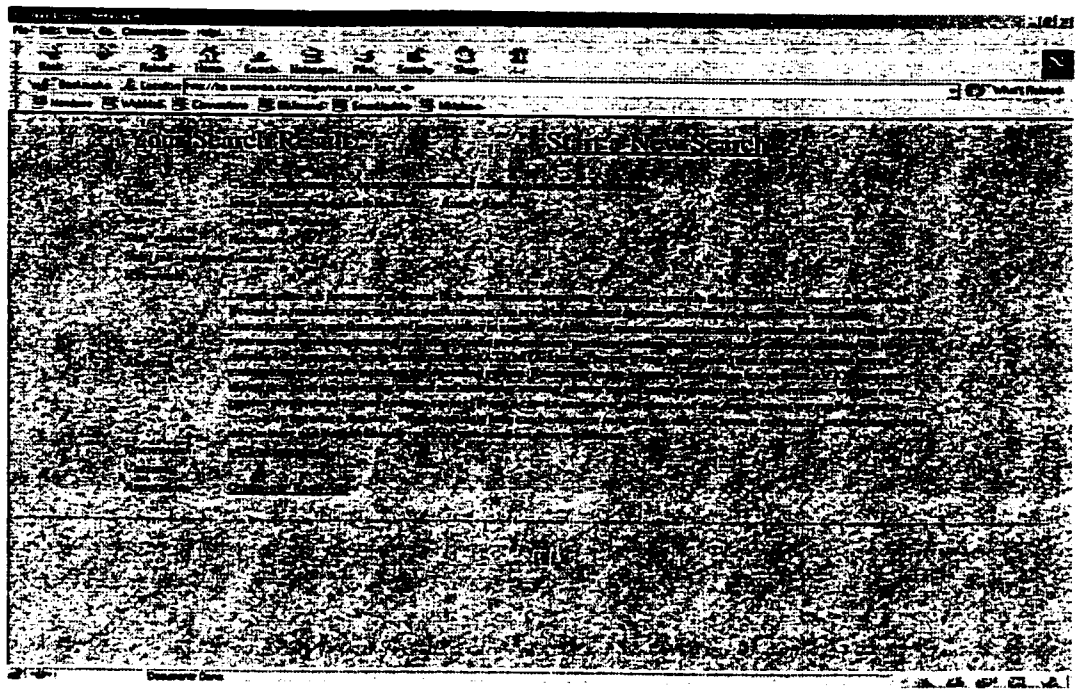

Figure 5.1.1.3 Search Result Interface

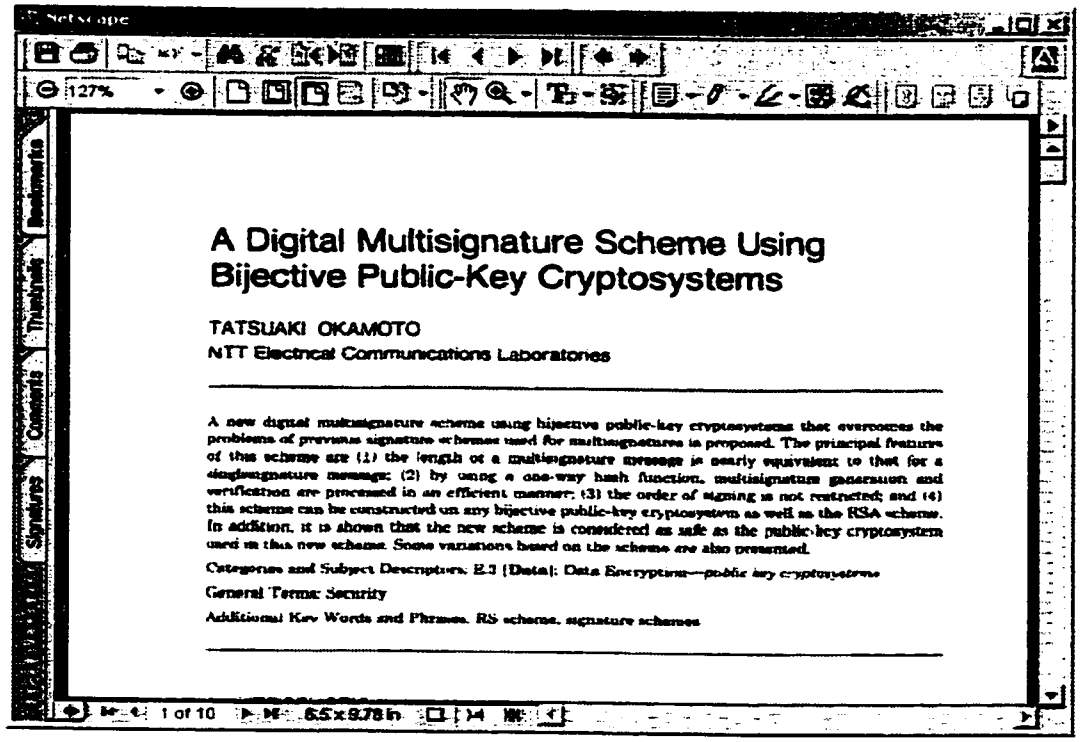

Figure 5.1.1.4 Resource Showing Interface

5.1.2 Query Structure of the Search Sub-system

The search query structure can be described by the grammar written in BNF as given below.

<search> ::= <operand> [<op> <operand> [<op> <operand> [<op> <operand>]]]

<operand> ::= <title> | <subject> | <authorlist> | <keywordlist>| <create_date>

<title> ::= <exacttitle> | <substringtitle>

<subject> ::= <general> | <general> AND <sub-subject> | <general> AND <sub-subject> AND <sub-sub-subject>

<authorlist> ::= <author> | <AND | OR> <authorlist>

<keywordlist> ::= <keyword> | <AND | OR> <keywordlist>

<create_date> ::= <fromdate> | <todate> | <fromdate> AND <todate>

<fromdate> ::= <day> AND <month> AND <year>

<todate> ::= <day> AND <month> AND <year>

<op> ::= AND | OR

<exacttitle> ::= <string>

<substringtitle> ::= <string>

<subject> ::= <string>

<sub-subject> ::= <string>

<sub-sub-subject> ::= <string>

<author> ::= <string>

<keyword> ::= <string>

<string> ::= <character> | <character> <string>

<character> ::= a | b | c | d | ... | y | z | A | B | C | ... | Y | Z | 0 | 1 | ... | 8 | 9 |

<day> ::= 1 | 2 | 3 | ... | 29 | 30 | 31

<month> ::= "January" | "February" | ... | "November" | "December"

<year> ::= 1990 | 1991 | ... | 2009 | 2010

We can see the query structure contains "title" or/and "subject" or/and "authorlist" or/and "keywordlist" or/and "create_date". The system will search the resource with the exact title, which users input; or the resource with the title that has a sub-string, which matches the user's input. The search criteria can also have a given subject, or a given subject and a given sub-subject, or a given subject, given sub-subject and given sub-sub-subject. Moreover, the search can be conducted with the author list. The search may involve an author list, or one of the authors of the resource. The search can also query with the one or some keywords of the resource. Also, the search can query with the period of the resources create date. Furthermore, the search can query with the combination of all these fields by "AND" or "OR" operator.

5.1.3 Error Handling of Search Sub-system

A search is based on the value of fields that users enter from the search interface. Before sending the query to the database, the system checks the query for validity. The CINDI system checks and makes sure that there is at least one field that has been entered. If none

of them have been entered, the system will give an error message to its users (figure 5.1.3.1).

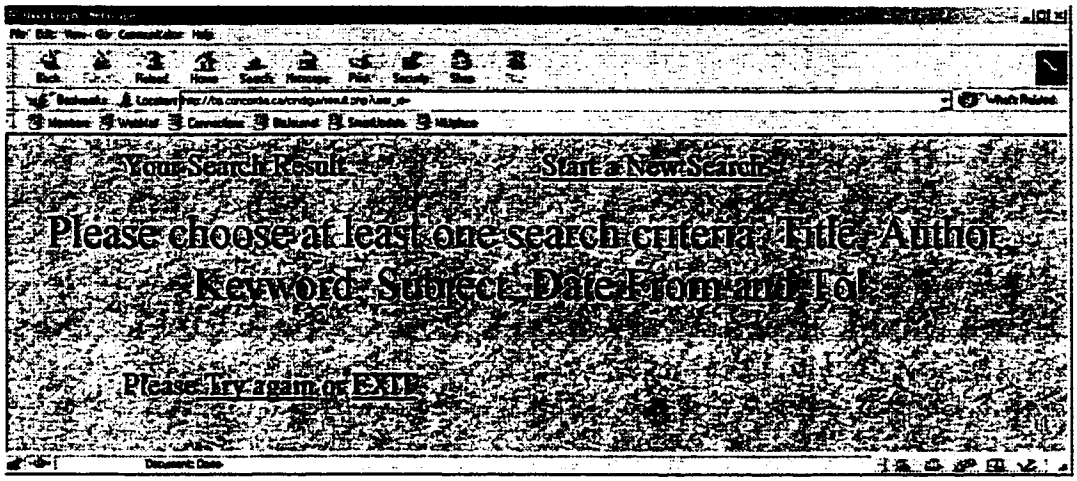

Figure 5.1.3.1 Query Error of No Search Criteria

The system also checks for any logical errors in the query. For example, if the title is empty but users choose "AND" as operator, the system will give users an error message to inform the users. Figure 5.1.3.2 shows the error message where a user did not input the title value but chose "AND" operator with the author field.

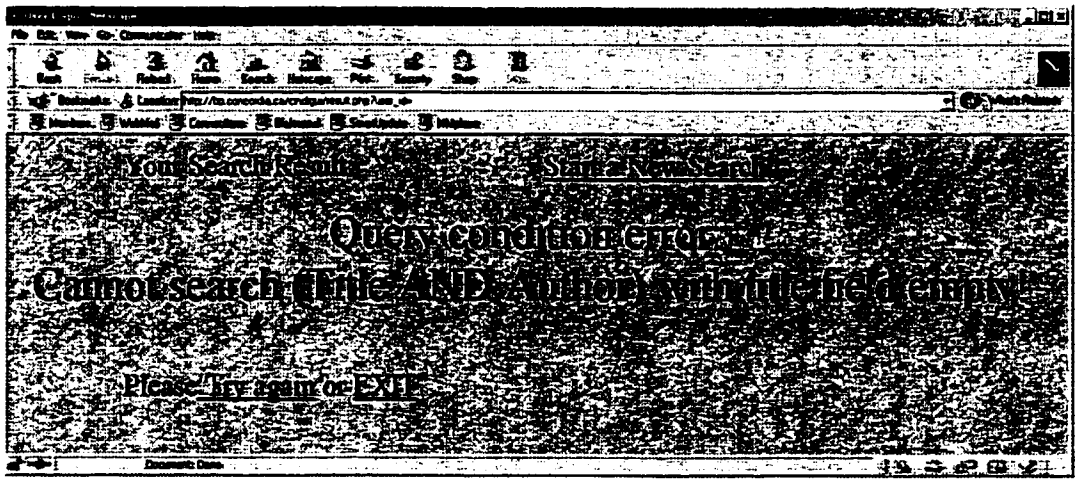

Figure 5.1.3.2 Query Error of Query Condition Error

5.2 Annotation

The annotation sub-system allows a user to comment on a resource in the CINDI system. Users can click the "annotation" link to read the annotation of other people for a resource, and they may add their own comments. The link is as shown in Figure 5.1.1.3. After clicking this link, the annotation page is shown to users as in Figure 5.2.

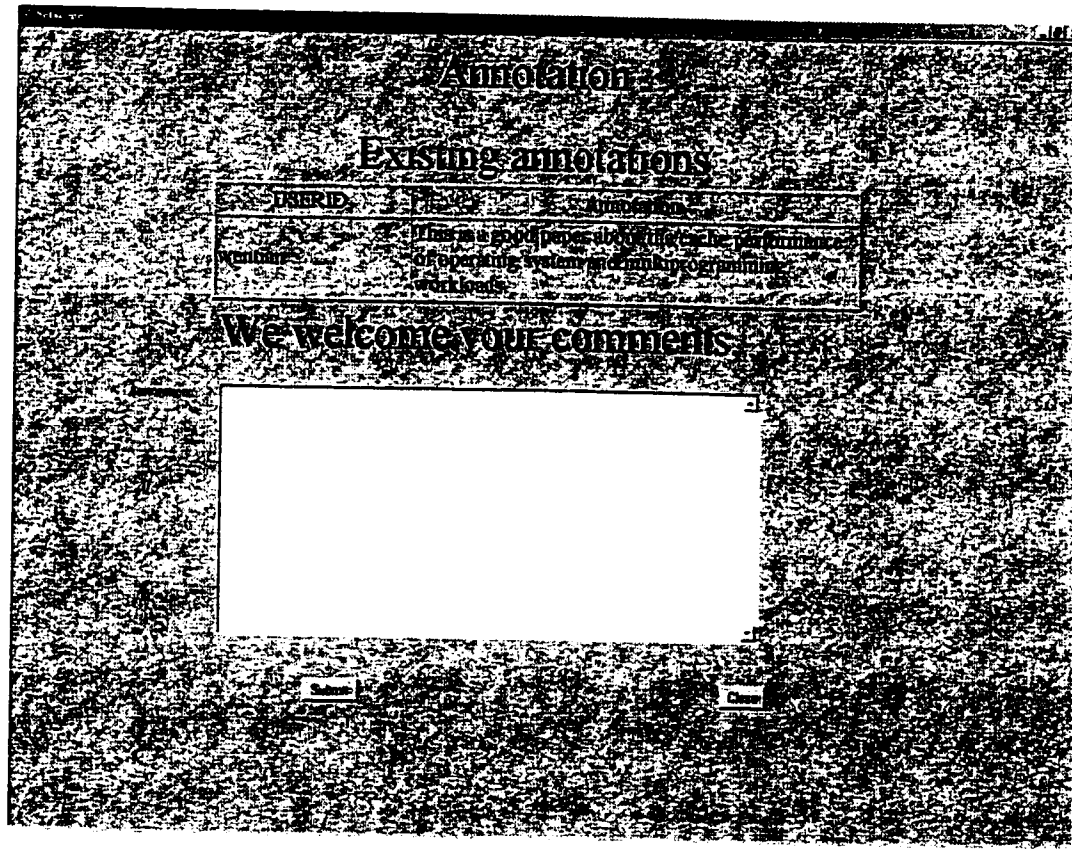

Figure 5.2 User Interface of Annotation Sub-system

The system can show all the comments about this resource made by other users. Users can also input their own annotation for this resource in the annotation field. The system will use the user login ID to store the user personal information and the annotation into the database, and the users need not input their name and personal information again.

Chapter6

Conclusion and Future Work

6.1 Conclusion

Almost all the existing virtual libraries just simply collect the information, and index them by their own indexing systems. This scheme has several drawbacks. One of them is lack of standardization. Since different people may have different perspectives on the same concept, the same document may be indexed into different category in different virtual library indexing systems. As a result, users of the virtual libraries may not find the information they need in the catalogue they think the document belongs to. Moreover, most of these virtual libraries search the information only using keywords. This scheme can cause lots of miss-hits since the same words may have different meanings in different domains and different places. For example, in Google when you search using a word, it may return thousands of hits. However, only a few of these may be relevant.

The CINDI system is an on-going project under the supervision of Dr. Desai. Its purpose is to develop an efficient and effective virtual library by using a standard index system, used in the Semantic Header.

The CINDI system tries to address the problems in two ways. One is by using the Semantic Header to achieve standardization. This meta-information is provided by the contributors of the documents or by the ASHG (Automatic Semantic Header Generator) of the system. Since the provider of the document provides the meta-information, the document should be indexed into more precise catalogues. Furthermore, this meta-information-for example, the subject groups-is based on the standard used in such domains of knowledge. For example, for the Computer Science subject hierarchy, we follow the ACM indexing system. For the Electrical Engineering subject hierarchy, the indexing is based on INSPEC.

For the users of the CINDI system, user-friendly interfaces are provided. In this interface, users can input the title, keyword, and author(s) of the documents searched. Also, the system provides a three-level hierarchy subject pull-down menu, and lets users choose the hierarchy subject group the documents belong to. This reduces potential miss-hits.

6.2 Contribution of this report

The contribution of this report falls into two areas: the system architecture and, the user-friendly graphic user interface.

- **The system architecture**

  In this report, we employ Three-Tier architecture for CINDI system. We store both the SHDB (Semantic Header database) and Catalog database into one server. This can

really increase the speed of the system, and avoid unnecessary network traffic. The 3-tier architecture of the CINDI system is shown in Figure 6.2.1.

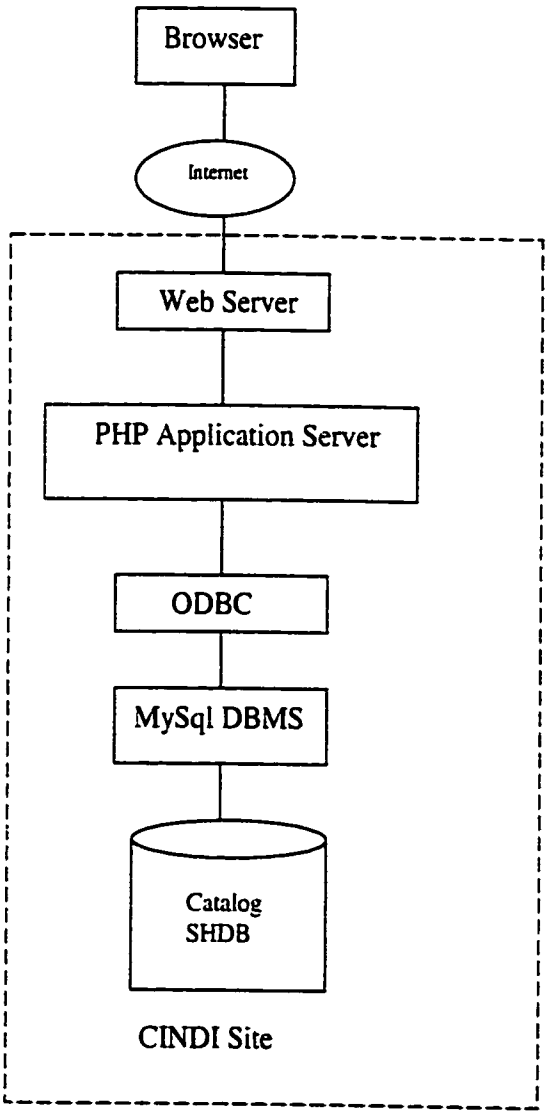

Figure 6.2.1 Three-Tier System Architecture for CINDI system

- **The user-friendly GUI**

  The other contribution of this report is the user-friendly GUI, especially the three-level hierarchy subject pull-down menu. Since the main contribution of the proposal of the CINDI system is standard indexing, cataloging documents into correct standard catalogue is very important. If the user interface asks the users to input the subject, sub-subject, sub-sub-subject name for the resources, different users will input different names even though they mean the same thing. The pull-down menu can resolve this problem very well. It lets users choose the subject they want and the system won't have the spelling errors and the non-standard subject names. These subject names in the pull-down menu are based on expert standards in their field, and are stored in the Catalog database. Moreover, this subject pull-down menu group is hierarchical. This means that once the users choose the higher-level subject, the lower-level subject pull-down menu will just show that the subjects belong to the higher subject domain. This can avoid inexperienced users from choosing unmatched subject groups.

6.3 Future work

The basic components of the CINDI system have been implemented. However, there are several things that can be improved in the future.

- **Separate the web server and database.**

  As an experienced web application developer, it is well known that it is insecure to put the database in the same machine as the web server. A web server is inherently a

publicly accessible machine, and should only contain information that either needs to be provided to the public or has recently been collected from the public. Therefore, separating the web server and the database is a smart approach for future projects for the CINDI system.

- **Add survey function to collect user information.**

  Collecting users' information is very important to a web application. If you get familiar with the users of your system, you can make your system more suitable to their requirements. As is well known, not many people will write to complain about an unfriendly web site. If the system doesn't fit their needs, users will just leave and not use the system. So being user friendly is very important for a web application. Collecting users' information can catalog the users into different group. As a result, the system can have its special solution for users in different groups. For example, for users using different languages and from different cultures, the system can have different user interfaces. So this survey function, in fact, is the prerequisite for internationalizing the CINDI system.

- **Internationalizing**

  Internationalizing is very important to a web application. Since the users of the system may come from different countries, they may use different languages, and they may have different cultural backgrounds. Without a doubt, users like to have a system using their native language, and matching their culture. One of the future works of the CINDI system would be to make CINDI international. Different kinds of

users can then use their own language to access CINDI, and get documents in their own language.

- **Online Help**

  In the current CINDI system, online help for users was not implemented. In fact, this is very important. Online help can provide users with better understanding of the system. Users would know what the system does, how to use the system and what the system needs as input. On line help can help users avoid mistakes. For example, there is a field in Semantic Header, named Coverage. Users may not know what this Coverage field means, and they might not know what they should input in this field. If there is an on line help, users can check and know what the system expects.

Bibliography

[BCD 95] Bipin. C. Desai,

The semantic Header Indexing and Serching on the internet

Computer Science, February 1995

[BISO 00] Ann P. Bishop, Clifford Lynch, et al

Digital library use (panel session): social practice in design and evaluation

Proceedings of the fifth ACM conference on ACM 2000 digital libraries

June 2 - 7, 2000, San Antonio, TX USA, Pages 276-277

[DAVI 98] David Medinets

PHP3 Programming Browser-based Applications

Mcgraw-Hill, 1998

[Fung 95] Fung R. and Del Favero B.

Applying Bayesian Networks to Information Retrieval

Communication of the ACM, Vol38, No.3, page 42-57

March 1995

[GEHA 86] Gehani, Narain and McGettrick, Andrew.

Software specification techniques

Workingham, England; Reading, Mass:Addison-Wesley,1986

[KATZ] Katz, W. A.,

Introduction to Reference Work,

Vol.1-2 McGraw-Hill, New York, NY

[Moha 01] Mohamed Amokrane Mechouet

WEB BASED CINDI SYSTEM,

Pages 19, Figure 4

Master thesis of Computer Science

Concordia University, April 2001

[Nono 91] Hajime Nonogaki and Hirotada Ueda.

FRIEND21 project: A Construction of 21st Century Human Interface.

In Human Factors in Computing Systems, pages 407-414.

Proceedings SIGCHI'91, New Orleans, LA, April, 1991.

[RBPE 91] Rumbaugh, James, Blaha, Michaeland et al

Object-Oriented Modeling and Design.

Prentice Hall, 1991

[SHAY 97] Shayan N., CINDI: Concordia Indexing and Discovery system,

Master thesis of Computer Science

Concordia University, 1997

[UML 98] Grady Broch, James Ruimbeaugh, Tvar Jacobson
The Unified Modeling Language User Guide
Addison-Wesley Pub Co, 1998

[Well 00] Luke Welling, Laura Thomson
PHP and MySQL Web Development
SAMS, 2001

[ZHOU 97] Youquan Zhou,
CINDI: The Virtual Library
Master thesis of Computer Science
Concordia University, 1997