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.

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

PORTING THE AUTOMATIC SEMANTIC HEADER GENERATOR TO THE WEB

ZHAN ZHANG

A MAJOR REPORT

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

AUGUST 2002

Abstract

Porting The Automatic Semantic Header Generator To The Web

Zhan ZHANG

The Concordia INdexing and DIscovery system (CINDI) is an indexing system. It enables a user to index and discover information resources from the CINDI virtual library. The information resource is described by using a meta-data called a Semantic Header. Automatic Semantic Header Generator (ASHG) is an automatic tool for the extraction and storage of some of the meta- information in a Semantic Header and an automatic text classification scheme.

This major report describes how to port the ASHG to web server on Linux. It is part of the work to develop a Web-based CINDI system. In the web based ASHG system, MySQL is employed for storing the ASHG's thesaurus. Apache is used as web server. The PHP script language is also used to create the user interface. All functions of ASHG are developed by using C++ and Perl.

The functions of the extraction of the meta-data from the existing ASHG are ported and the main algorithms of ASHG are adapted to the web-based system. We redesigned and implemented the ASHG's architecture, the web-based user interface, the ASHG's thesaurus, the programs used to build and maintain the thesaurus and all interfaces between main algorithms and database in the web based ASHG system.

Finally, the ASHG system has been integrated with Web-based CINDI system.

Acknowledgements

I would like to thank my supervisor, Dr. Bipin C. Desai, for his patience and valuable guidance over a period of one and half years. Without his enthusiastic technical support in the system environment, this work would not have been possible.

I would like to sincerely thank Dr. J. William Atwood for his guidance and help.

I am sincerely grateful to Yuhui Wang, Wen Tian, and Xiaomei Yang who, as part of the CINDI project, suggested many ideas to me during our valuable discussions.

Thanks also should go to Sami Samir Haddad and Abdelbaset Ali who were the previous developers for this project and some functions are based on their development.

Finally, I would like to dedicate this work to my wife, Min Huang and my two lovely daughters, for their understanding, continuous support and encouragement.

Contents

1 Introduction 1

1.1 Information retrieval system 1
1.2 CINDI 2
1.3 Meta-data 3
1.4 The aim of this project 4

2 The Analysis of ASHG 5

2.1 Web-based CINDI and ASHG 5
2.1.1 Overview of the CINDI 5
2.1.2 Web-based CINDI 6
2.1.3 ASHG 8
2.2 The problems of porting 9
2.3 Porting the ASHG to the web 10
2.3.1 User requirements of web-based ASHG 10
2.3.2 Porting requirements 10

3 ASHG New Environment 12

3.1 New working platform 12
3.2 Languages 13
3.3 Databases 14
3.4 Interface 15
3.5 Architecture 17

4 Porting ASHG's Thesaurus 19

4.1 The Thesaurus for ASHG 19
4.2 The Subject Hierarchies 19
4.3 The Controlled Term 20
4.4 The Control Term Subject Association 21
4.5 The implementation of the Thesaurus 22
4.5.1 The subject headings 22
4.5.2 The Control Term 's class 25
4.5.3 Keyword class 27
4.5.4 The table schemas of the Thesaurus 28
4.5.5 Physical implementation of the tables 30
4.6 Programs used to maintain the Thesaurus 32

5 Porting ASHG 34

5.1 Introduction 34
5.2 Document Type Recognition 35
5.3 Applying ASHG's Extractors 38
5.4 ASHG's Document Subject Headings Classification scheme 40
5.5 Semantic Header Validation 41

6 Integrating and Testing 44

6.1 Integrating 44

6.1.1 Multiple Users vs. File Name and Type 44
6.1.2 Intermediate tables 45
6.1.3 Security 46
6.2 Test Results 47
6.3 The analysis of Test Results 53
6.4 Sample Results 57

7 Conclusion and Future Work 61

7.1 Conclusions 61
7.2 Future Work 62

Appendix 64

References 66

List of Figures

Figure 1: The architecture of the ASHG system 18
Figure 2: The classes of three level subject headings 23
Figure 3: The classes of control terms 25
Figure 4: The class of Keyword 28
Figure 5: Uploading page 36
Figure 6: File type confirmation page 38
Figure 7: Semantic Header Validation 42
Figure 8: Subject Heading Validation 43

List of Tables

Table 1:HTML test results 49
Table 2:Latex test results 50
Table 3:Text test results 51
Table 4:RTF test results 52

Chapter 1

Introduction

1.1 Information retrieval system

Information Retrieval (IR) is concerned with the representation, storage, organization and access of information [5]. The primary goal of an IR system is to retrieve all the documents, which are relevant to a query while retrieving as few non-relevant documents as possible.

An IR system always deals with natural language text, which is not always well structured and could be semantically ambiguous, and has to understand the contents of the information resource. This understanding of document content involves extracting syntactic and semantic information from the document text and using this information to match the query. The difficulty is not only knowing how to extract this information but also knowing how to use it to decide relevance.

A typical IR system should have three components: input, processor and output. Firstly, the main problem of input side is to obtain a representation of each document and query suitable for a computer to use. Most IR systems store only a representation of the documents, which means that the text of a document is lost once it has been processed to generate its corresponding representation. A document representation could, for example, be a list of extracted words considered significant. Instead of having computer process natural language, an alternative approach is to employ an artificial language to formulate

all queries and documents, which could be more likely effective provided that the user is willing to express his information with the language. The best representation should avoid problems caused by different semantics and incomplete or incorrect data cataloguing. Secondly, the processor of the retrieval system is concerned with the retrieval process, which may involve structuring the information in some appropriate ways, such as classifying it. It will also involve performing the actual retrieval function, i.e., executing the search strategy in response to a query. Finally, the IR system output generates a set of displays or document numbers.

Generally, database, documents representation, and retrieval process are key technical points in an information retrieval system. The notion of relevance is at the center of information retrieval.

1.2 CINDI

The CINDI is an information retrieval and index system, proposed by Dr. Desai [6]. The objective of the project is to build a system that enables any resource contributor to catalog his/her own resource and any user to search for hypermedia documents using typical search criteria such as Author, Title, Subject, etc. The system will offer a bibliographic database that provides information about documents available on the Internet. It will provide a mechanism to register, search and manage the meta-data. A standardized index scheme (Semantic Header) will be used to ensure homogeneity of the syntax and semantics of such an index. These index entries are stored in a database system called the Semantic Header Database System. In addition the catalog for subjects will provide help information when the user catalogs his/her index.

1.3 Meta-data

In simplistic terms, meta-data is data about data. Meta-data allows us to identify the meaning, context and validity of data. Meta-data provides a set of facts about the structure, organization and behavior of a given set of data.

Meta-data can be defined as additional data associated with some file or document. While the meta-data might appear within the document itself (such as a list of keywords), the idea is that the meta-data can be considered separately from the parent document.

The term meta-data has different connotations depending on the field of application. From an information management perspective, meta-data may refer to the attribute descriptors applied from standards such as XML schemas. On the other hand, meta-data, may refer to attributes required for data integration and transfer activities involved in a data warehouse project.

By semantic meta-data we denote additional information that is used to describe the meaning of information. In the CINDI system, we use meta-data called the Semantic Header to describe the semantic contents of the information resource.

There are two kinds of tools to handle meta-data in information retrieval system: automatic meta-data generation and manual input meta-data tools. Some of the meta-information can be generated automatically for collections of text documents (date, size and type of file). Some of the meta-data are difficult or impossible to extract automatically, such as quality of the resource, especially from collections of multimedia.

1.4 The aim of this project

As part of the CINDI system, the ASHG is an automatic tool for the extraction and storage of some of the meta-data in a Semantic Header and an automatic text classification scheme. The Semantic Header is stored in the CINDI system. This meta-data could be entered either by the information resource provider or by the ASHG. The ASHG generates and extracts part of the meta-data (Semantic Header) of the submitted document, assists the user in this process, and classifies the resource under a list of subject headings. It will help the provider to verify and correct the Semantic Header entry.

The existing ASHG has been implemented under the Unix system using the Motif toolkit, Perl, C programming language. Object Database and Environment (ODE) is the database system used for the ASHG's thesaurus.

This major report describes the work and effort of porting the ASHG to web server on Linux, which is part of the work to develop a Web-based CINDI system. In the web-based ASHG system, MySQL is employed for storing the ASHG's thesaurus and Apache is used as web server. The PHP script language is also used to create the user interface. All of functions of the ASHG are developed in C++ and Perl.

The functions of the extraction of the meta-data from the existing ASHG are ported; the main algorithms of the ASHG are adapted to the web-based system. We have redesigned and implemented the ASHG's architecture, the web-based user interface, the ASHG's thesaurus, the programs used to build and maintain the thesaurus and all interfaces between main algorithms and database in the web-based ASHG system.

Chapter 2

The Analysis of ASHG

The ASHG has been developed by Haddad under the Unix system, and the user interface has been implemented using the Motif toolkit and C language [8]. We will call this ASHG the Motif-based ASHG, the new ASHG as the web-based ASHG.

2.1 Web-based CINDI and ASHG

The web-based ASHG will be integrated with the web-based CINDI system.

2.1.1 Overview of the CINDI

The Internet provides much more than access to electronic mail. A major scope of the Internet is information retrieval. Thousands of host institutions have made their material electronically accessible to interested users. A variety of different data and formats was used by different host.

The IR system will be needed for easy search for and access to resources available on the Internet, and will be distributed information system. Even though under control of a single administrative unit, it may have multiple problems typically caused by differences in semantics and representation, and incomplete and incorrect data dictionaries. Building a standard index structure and a bibliographic system using standardized control definitions and terms can solve these problems and lead to a fast, efficient and easy

access to the documents. Such definitions could be built into the knowledge base of expert system based index entry and search interfaces [2].

The problem with the indices of many existing search systems is that their selectivity of documents is often poor [2]. Because of poor choice of search terms, there is a big chance of getting correct document but missing relevant information. In addition, the user is required to access the actual resource, based just on the title and author information as provided through a library catalog, and decide whether the resource meets the needs. These problems are addressed by the CINDI. The CINDI uses an index entry called Semantic Header, which includes items such as title, abstract, keywords, and subject, and provides a mechanism to register, manage and search the bibliography.

The overall CINDI system uses knowledge bases and expert sub-systems to help the user in the register and search process. The expert system will help the user in entering those attributes required for indexing or updating a Semantic Header. The CINDI standardizes the terms. The index generation and maintenance sub-system uses CINDI's thesaurus to help the provider of the resource select correct terms for items such as subject, sub-subject and keywords. Similarly, another expert sub-system is used to help the user in the search for appropriate information resources [1].

2.1.2 Web-based CINDI

The original CINDI system was designed as a client/server application system under the Unix system. It is a traditional two-tier model. This type of architecture is suitable for deployment to a small group of users, but has many problems in a large-scale application system.

The CINDI system was designed to require two important sub-systems. A registering system is required to register the Semantic Header into the distributed Semantic Header database (SHDDB), and a search system is required to allow users to enter a query based on multiple fields. The register and search sub-systems are to be carried out on the WWW.

The web-based CINDI system is designed for the web environment. It would have the following functions:

- A standardized metadata (Semantic Header) to describe each information resource.
- A Semantic Header Database that stores the Semantic Headers.
- A Semantic Header registering system.
- A search system that allows querying entry.
- An annotation system.
- Subject tables that store information about subjects classified using a standard cataloging scheme.
- ASHG subsystem to generate the Semantic Header.
- ASHG's Thesaurus.
- Security Control sub-system.
- Uploading resource.
- Web-based user interfaces.

2.1.3 ASHG

The Motif-based ASHG was also designed as a client/server application system under the Unix system. It was designed to require two sub-systems. ASHG's thesaurus is required to store the three level subject headings and a set of control terms. An automatic tool allows users to extract and store some of the meta-data in a Semantic Header and classify the document into a set of subject headings. The user interface was implemented using Motif and C language as client side. The main functions of the ASHG and ASHG's thesaurus were developed as server side by using Perl language, the O++ database programming language, and The Object Database and Environment (ODE). The thesaurus is defined, queried and manipulated using O++, an extension of C++. A few facilities have been added to C++ to make it suitable for database applications. O++ provides facilities for creating persistent objects, which are stored in the database and for querying the database. The thesaurus, itself, is also based on the client-server architecture. Communication between the client and server has been implemented using the TCP/IP protocol.

For web-based ASHG, we need to develop the following functions:

- ASHG subsystem to generate the Semantic Header.
- ASHG's Thesaurus.
- Uploading resource.
- Web-based user interfaces.
- Integrating the web-based ASHG with the web-based CINDI.

2.2 The problems of porting

Actually, this project is about porting an application from Unix (Solaris) to Linux. Generally, if an application has been implemented with standard programming interfaces, the porting process is supposed to be quite simple. Because Solaris is a certified Unix implementation, it has passed the conformance tests of the Unix copyright holders. Linux is also designed with conformance to the Unix standard. If we use only the set of interfaces covered by the Unix standard, we can reuse the code without any changes.

The ASHG is composed of several programs and database. We not only need to port the programs but also change the architecture of the ASHG. The porting problems that we can expect occur in several different areas:

1. The tools used on the different platforms are different. This can introduce additional problems beyond the differences in the use of the tools. The languages the compilers accept are slightly different.
2. The programming interfaces are heterogeneous. While both operating systems are designed to follow the respective standards, the differences in a variety of the system implementations are unavoidable. The programming environment is regulated by a common standard, but there is still room for differences and extensions.
3. The architectures are distinct. The Motif-based ASHG was designed as a client/server application system, whereas the web-based ASHG was a three-tier web application, known as presentation layer, business layer and database layer.

4. The users of the Web-based system have different permissions compared to regular users of the system. Some system functions may not be used in the application.

According on these differences, we need to find all available problems for porting in detail and give suitable solutions in new environment.

2.3 Porting the ASHG to the web

2.3.1 User requirements of web-based ASHG

To use the web-based ASHG, the user can use web browsers to access the ASHG. First the user is asked to upload the document from their local machine to the web server, and then the web-based ASHG will recognize the type of the document uploaded. The type result will be showed to the user. The next step is the user applies the extraction function of the ASHG on the document to generate the Semantic Header, which includes a list of subject headings depending on the ASHG's thesaurus. Finally, user will confirm, verify and modify this Semantic Header. If satisfied, the user will register it to the Semantic Header database of the CINDI.

2.3.2 Porting requirements

The uploading function is a new function that we want to add to the web-based ASHG. We need to redesign the architecture of the Motif-based ASHG to fit the web server on Linux and the ASHG's thesaurus. For the Motif-ASHG, the functions of the extraction of the meta-data that were implemented in Perl script language are adapted to the web-

based ASHG, but some system functions will be changed in Perl programs. The main algorithms of the Motif-based ASHG are used to the web-based ASHG, but will be implemented using C++ to replace O++. The programs used to build and maintain the thesaurus are rewritten in C/C++. All interfaces involved defining, manipulating and querying the database are also rewritten in C++. O++ and ODE used in the Motif-based ASHG will be replaced in the web-based ASHG by MySQL and the new user interfaces use web pages.

Chapter 3

ASHG New Environment

3.1 New working platform

The user of the CINDI and ASHG will use web browsers (Internet Explorer or Netscape) as user interface on any operating system, such as Mac OS, Windows, Unix or Linux. For the server side we choose Linux as the platform of the ASHG system because it is free and its compatibility with the implementation of the Motif-based ASHG in Unix. This server should have high security and reliability. In the case of Linux, it has reliability and security that comes with UNIX.

Since this is a web-based application, a dedicated web server will be needed. For the web server security reason, we adopt Apache as the web server since is a powerful, flexible, HTTP/1.1 compliant web server.

The ASHG is a multiple-user application and also has a thesaurus. The database server will be needed. MySQL is used as database server in this application because it is free and is suitable for middle size application.

The user interface of this application will be web pages, and uploading users' resource from client side to server side also will be implemented on web pages. In the web pages, it includes not only HTML page but also dynamic HTML format. So a server-side HTML-embedded scripting language will be needed.

PHP can do anything any other CGI program can do, such as collecting form data, generating dynamic page content, or sending and receiving cookies. PHP provides an

easy way to upload documents also. PHP has support for talking to other services using protocols such as IMAP, SNMP, NNTP, POP3, HTTP and countless others.

Finally we use HTML and PHP to build web pages. It is so easy to write dynamic web pages using PHP. By the way, it is pretty easy to connect MySQL with PHP.

3.2 Languages

The ASHG uses four languages, Perl, C/C++, PHP, HTML. PHP and HTML were used to build web pages. Perl language was used to handle information resource. C/C++ language was used to represent subject headings, store Semantic Header into database and maintain database.

PHP is a kind of server side script language, and is commonly said to be faster and more efficient for complex programming tasks. We use it just for the user interface of web pages. PHP is also needed to connect to MySQL: this connection is between the user interface and database, and it always is needed when the user interface requires to the database of Semantic Header.

Perl is a high-level, multi-purpose public domain language that has established itself as the scripting language of choice - replacing facilities such as the shell, sed and awk. Perl has powerful text-manipulation functions. It eclectically combines features and purposes of many command languages. Perl has enjoyed recent popularity for programming World Wide Web electronic forms and generally as a gateway between systems, databases, and users. It has been used with good success for systems administration tasks on Unix systems.

Perl is used for many diverse tasks. But in this application, it was not used for systems administration tasks in Linux or as CGI programming and web pages script language. It is a programming language, which excels at handling textual information and good at using the Linux system utilities. All functions that extract meta-information from information resource were written in Perl.

For C/C++ language, we do want to use the Object Oriented features of C++. The ASHG's Thesaurus is composed of a three level subject hierarchies. We can represent the subject hierarchy using Object Oriented programming with C++. All the interface APIs that are between ASHG extraction functions and MySQL database were written in C\C++. The different function modules of ASHG and other programs of CINDI were integrated using C\C++ also.

3.3 Databases

For this web application, a dedicated database server is needed. MySQL is a compact database server. In addition to supporting standard SQL, it compiles on a number of platforms and has multithreading abilities on Unix servers, making it achieve better performance from the underlying system.

MySQL database server is a true multi-user, multi-threaded SQL database server, freely available, open-source, actively maintained, and a client/server implementation that consists of a server daemon mysqld and many different client programs and libraries.

In web application, MySQL supports many simultaneous users; each MySQL client gets a dedicated thread in the MySQL server, which allows different users to access the same

tables at the same time. All MySQL operations are atomic: no other users can change the result for a running query.

MySQL server is fast, reliable, and easy to use. MySQL also has a very practical set of features developed in close cooperation with users.

The MySQL server is coded mainly by a single person with many years of coding experience, very little redundant code is in it. Most of the basic algorithms also come from an era of slow CPUs and small amounts of memory. The algorithms have mostly been extended to use larger caches if there is available memory. As a result, MySQL has a compact fast design, the code size of the server is less than 1MB on an i386, which normally uses very little memory, but can be configured to take advantage of large amounts of memory.

Though under constant development, today MySQL offers a rich and very useful set of functions. The connectivity, speed, and security make MySQL highly suited for accessing databases on the Internet.

MySQL database is a very popular Open Source database, and it uses SQL92, a milestone in database query language, as a standard. MySQL has robust interfaces for C/C++, Perl and PHP.

3.4 Interface

The user of the ASHG uses web pages as user interface through the Browser. Dynamic HTML page technology is used in this application. The other function to be added is the ability to upload information resource by users. The ASHG allows the users to upload a

file from client side to server side. After uploading the file to server side, the ASHG will automatically recognize the document type and generate the Semantic Header of this file. File uploading is a process of sending an arbitrary file from a client machine to the server. For uploading file, web-based file upload is vastly superior alternative to other means of transferring files to a server over the Internet protocols.

FTP has been the standard mechanism for sending files to a server since the earliest days of TCP/IP. It is reliable, can take into account text vs. binary files across platforms. However, with FTP uploads you must either manage many user accounts or allow anonymous access. Some organizations do not allow FTP for security and intellectual property reasons.

With a web application, you can limit the size of the uploaded file dynamically every time it is invoked. You could even change the size depending on information contained in the form. Additionally, you can flush upload that do match certain criteria, such as wrong file type or file contents. A pleasing web page can offer instructions, advise, on-line help. This is not possible with batch based FTP. More importantly, when errors occur, you can provide immediate feedback to the user and offer corrective action.

Web-based uploading is performed via an HTML form with the attribute ENCTYPE="multipart/form-data". This form must also contain one or more <INPUT TYPE=FILE> items with which the user specifies local files to be uploaded. The data posted by a form with the ENCTYPE="multipart/form-data" attribute must be parsed by a server-side process to extract the uploaded files and other non-file items.

In this application, the form action points to a PHP script, and that script is a server-side process. PHP script checks that a file was sent, and then takes that file name (which is

associated with the file sitting in the temp directory) and executes a copy command, thus moving the file from the temp directory to a real live directory on the server. PHP script lets the user know that the file made it to its destination and all is well.

3.5 Architecture

The ASHG is a three-tier web application. It includes presentation layer, business layer and database layer. It is built by Linux, Apache, PHP, Perl, C/C++ and MySQL, is shown in figure 1.

The following is a brief explanation of this architecture.

Presentation layer includes all the user interface pages; it will call the functions of the ASHG and store in or retrieve Semantic Headers from the database of Semantic Headers depending on users' request.

Business layer is an ASHG sub-system. It extracts Semantic Header for HTML, Latex, Text and RTF documents. It provides the document's contributor an initial set of subject classifications and a number of components of the Semantic Header for the document.

Database layer involves two databases, ASHG's Thesaurus and Semantic Headers database. All initial subject headings, control terms associated with the subject headings and subject classifications are stored in the Thesaurus. All information including Semantic Headers, contributors/users personal information and user's annotation, etc. are stored in the Semantic Header database.

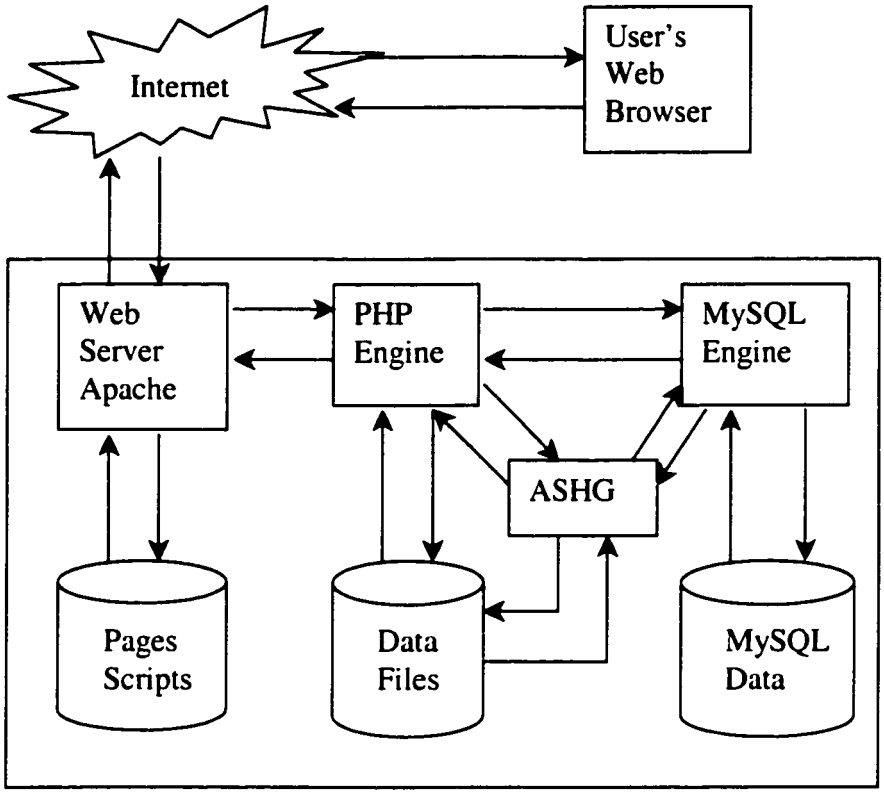

Figure 1: The architecture of the ASHG system

Chapter 4

Porting ASHG's Thesaurus

4.1 The Thesaurus for ASHG

The Thesaurus is a set of items (phrases or words) plus a set of relations between these items. The ASHG's Thesaurus is used to store the standardized subject headings, control terms and the control term subject association.

The ASHG's Thesaurus contains a three level subject hierarchy and a set of control terms associated with the subject headings found in the subject hierarchies. The Thesaurus used by ASHG consists of four object classes: *Level_0* represents the general subject of the subject hierarchy, *Level_1* represents the sub-subject of the general subject and is derived from *Level_0; Level_2* represents the sub-subject of the *Level_1* subject and is derived from *Level_1*, and finally control term which contains the root terms that are derived from the subject headings. A root term is the origin of all possible terms that can be generated from it by adding the suffixes and prefixes [8].

4.2 The Subject Hierarchies

For registering and searching the Semantic Header, the standardization of subject headings will be needed. Since different subject headings may be used to convey the same subject, and since different people may have different perspectives on the same single subject, controlled subject headings have been derived. The ASHG system focuses

on the standardization of subject headings. This database helps the provider of the information resource in selecting the correct subjects and sub-subjects' headings for the semantic header entry.

ASHG's subject hierarchy was based on classification scheme used by Association for Computing Machinery (ACM) and the Information Service for Physics, Electronics and Computing (INSPEC) initially, and refined using Library of Congress Subject Headings (LCSH). The resulting subject hierarchy was formed a three level hierarchy with one additional level. This last level contains terms used as control terms associated with the third level subject headings. ASHG's subject hierarchy is made up of three levels, where *Level_0* contains the general subject heading. Currently we have included only two general subject headings: Computer Science and Electrical Engineering. *Level_1* contains all the subjects that fall under *Level_0* subjects, like Software Engineering, and similarly *Level_2* will contain more precise subjects that fall under *Level_1* subjects, like Cost Estimation Management [8].

4.3 The Controlled Term

Some form of standardization of terms is also needed to avoid chaos introduced by differences in perception of different users. For each subject heading found in ASHG's subject hierarchy and the additional terms, we used their constituent English non noise words as their corresponding controlled terms. The subject headings found in ASHG's *Level_0, Level_1* and *Level_2* will be used as the basis for finding the controlled terms. In addition, the additional terms associated with ASHG's *Level_2* subject headings are mapped into controlled terms.

Mapping ASHG's subject headings terms into some controlled terms is discussed below:

1. English stop words are removed from ASHG's subject hierarchy headings and the additional terms associated with ASHG's *Level_2* subject headings.
2. Applying ASHG's stemming process to the remaining list of words in order to get their root, which will be stored in the list of controlled terms.
3. Generating a list of words to be added to the ADDED_WORDS_LIST file. These words are found in the subject headings but not in the *spell* check dictionary.

4.4 The Control Term Subject Association

The Control Term Subject association is an association between the controlled terms and their corresponding subject headings. The main reason behind the Control Term Subject association is to extract or classify the information resource under a number of subject headings by comparing the significant list of words contained in the document with the list of controlled terms.

Each controlled term is associated with one or more subject headings. It has three lists of subject headings attached to it. The three lists correspond to the general subject headings, sub-subject *Level_1* subject headings, and *Level_2* subject headings [8].

The association between the subject headings and the controlled terms is constructed by comparing the root words found in these subject headings with the ASHG's controlled term. A document often covers a number of subject or domains. ASHG uses the words in a document to classify it under a list of subject headings. This list of words from the document is matched against the controlled terms. If a match is found, then this subject heading is associated with the controlled term. The reason for building such an

association is that ASHG will generate a suggested list of subject headings using the words found in the document by consulting the Control Term Subject association. This process is represented below:

1. Split each subject heading and the terms associated with ASHG's *Level_2* subject headings into words they are made up of.
2. English noise words found in the list of words are removed.
3. Words are checked using the *spell* command.
4. Remove words neither in spell dictionary nor in the ADDED_WORDS_LIST file.
5. Apply the stemming process to generate the root-controlled terms from the words.
6. Each root-controlled term will be associated with the subject headings that contain it.

4.5 The implementation of the Thesaurus

In this application, C++ will be used to implement the logic of subject hierarchy and control terms. The classes will be represented using UML.

4.5.1 The subject headings

Since the three subject headings classes have similar characteristics, we will only describe the operation of the *Level_2* class. The operation *print()* prints *Level_2* object into the result file. The operation *print_all()* prints *Level_2* object as well as parent objects *Level_1* and *Level_0*. The operation *isLevel_2* return a *Level_2* object whose value is 'str', otherwise it return 0 or 1. The operation *list_all_level_2* prints into the

result file all *Level_2* objects where their *Level_0* and *Level_1* subject values correspond to 'lev_0' and 'lev_1'. The operation *get_lev_2* and *get_subject* return a pointer to the

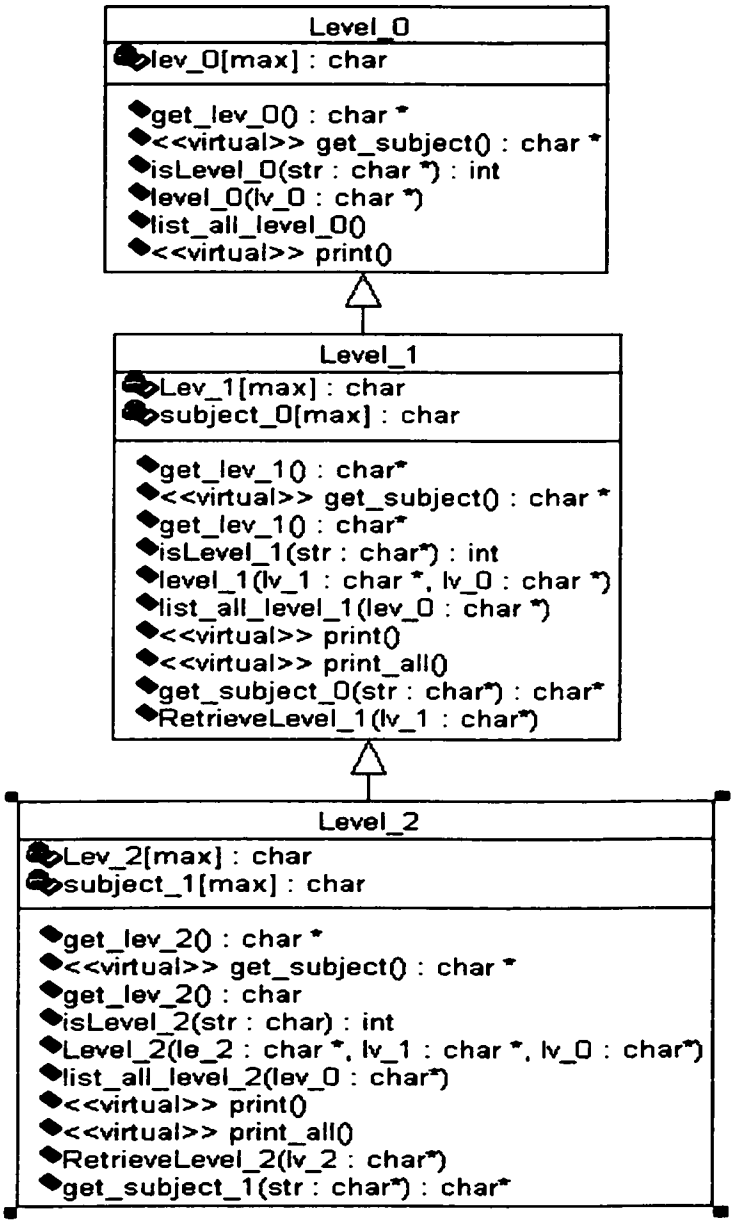

Figure 2: The classes of three level subject headings

Level_2 subject of the current object. Both are used in document classification. The operation *get_subject_1* returns a pointer to the *Level_1* object, which is the parent of a

Level_2 object whose value is the string 'str'. The constructor *Level_2* initializes *Level_2*'s and its parents' values. When a new object of class *Level_2* is created, it is an empty object. The operation *RetrieveLevel_2()* is similar to the constructor, but fills the *Level_2*'s and its parents' values to be obtained from the database. Its definition follows:

```
void Level_2::RetrieveLevel_2(char *lv_2)
{
if (isLevel_2(lv_2))
  {
   char mysqlstr[200];
   strcpy(mysqlstr, "SELECT L0.items, L1.items FROM level_0 L0, level_1 L1, level_2
L2  WHERE L0.level0=L1.level0 AND L1.level1=L2.level1 AND L2.items=LCASE('");
   strcat(mysqlstr, lv_2); strcat(mysqlstr, "')");

   res = mysql_query(&my_connection, mysqlstr);
   if (res) {
         printf("SELECT error: %s in Level_2 constructor\n", mysql_error (
&my_connection ) );
     } else
        {
          result = mysql_store_result(&my_connection);
          if (result) {
                /* begin */
                /* only fetch the first row if there are more than one rows.*/
            if (row = mysql_fetch_row(result))
                  {
                   strcpy(lev_0, row[0]);
                   strcpy(lev_1, row[1]);
                  }
                mysql_free_result(result);
                /* end */
             } else {
               if (mysql_errno(&my_connection)) {
                 fprintf(stderr, "Retrieve error: %s\n",mysql_error(&my_connection));
                 }
             } /* end if (result) */
        } /* end if (res) */
  } /* end of isLevel_2(lv_2) */
}
```

4.5.2 The Control Term 's class

There are three similar classes for *stack_array* for the three subject headings levels. We only describe the *stack_array_0* class. The operation *isempty* checks if array is empty.

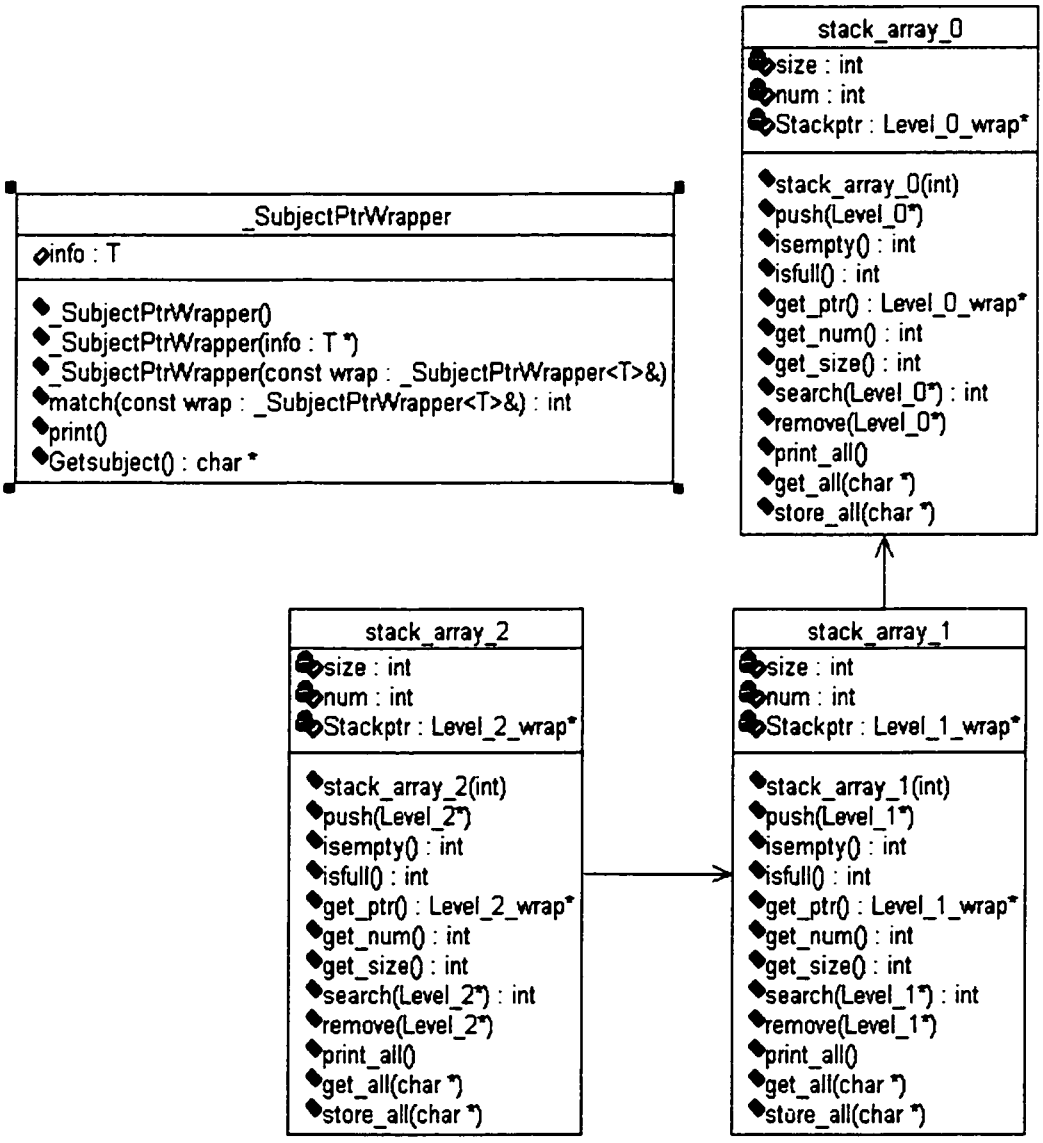

Figure 3: The classes of control terms

The operations *get_ptr*, *get_num*, and *get_size* are defined in the class description. The operation *push* adds a pointer to a subject heading into the array. If the array is full, then a new array of a bigger size is created, the old one is copied into the new one and the new pointer is then added to the array. The operation *isfull* checks if the array is full. The operation *search* looks for a pointer to a subject heading in the array. The *remove* operation removes a pointer to a subject heading from the array. The operation *print_all* prints all the subject headings pointed by in the array. The operation *get_all(s)* gets all the control term subjects pointed by the array of subject wrappers from the Thesaurus. Its definition follows:

```
void stack_array_0::get_all(char * s)
{
  Level_0 *ptr;
  char querystr[300];
  strcpy(querystr, "SELECT l.items FROM key_class k, lev_0_list l WHERE ");
  strcat(querystr, "k.idkey=l.idkey AND k.keyword=LCASE('");
  strcat(querystr, s); strcat(querystr, "')");
  res=mysql_query(&my_connection, querystr);
  if (res) {
     cout << "Get a error when select items from key_class and lev_0_list for: " << s <<
endl;
     exit(1);
  }
  result = mysql_store_result(&my_connection);
  if (result) {
    while(row = mysql_fetch_row(result)){
        ptr = new Level_0(row[0]);
      push(ptr);
    }
  } else {
        if (mysql_errno(&my_connection)) {
        cout << "Retrive error: in stack_array_0::get_all()" << endl;
        mysql_free_result(result);
          exit(1);
        }
  } /* end if (result) */
  mysql_free_result(result);
}
```

The operation *store_all*(s) stores all the control term subjects pointed by the array of subject wrappers into Thesaurus. Its definition follows:

```
void stack_array_0::store_all(char * s)
{
   char querystr[300];
   char id[20];
   strcpy(querystr, "SELECT idkey FROM key_class WHERE keyword=LCASE('");
   strcat(querystr, s); strcat(querystr, "')");
   res=mysql_query(&my_connection, querystr);
   if (res) {
      cout << "Get a error when select idkey from key_class for: " << s << endl;
      exit(1);
   }
   result = mysql_store_result(&my_connection);
   if (result) {
       row = mysql_fetch_row(result);
       strcpy(id,row[0]);
   } else {
         if (mysql_errno(&my_connection)) {
         cout << "Retrive error: in stack_array_0::store_all()" << endl;
         mysql_free_result(result);
            exit(1);
         }
   } /* end if (result) */
   mysql_free_result(result);
   for (int i=0; i<size; i++)
    if (Stackptr[i].info != '\0'){
          strcpy(querystr,"INSERT INTO lev_0_list(idkey, items) VALUES('");
       strcat(querystr, id); strcat(querystr, "','");
       strcat(querystr,  Stackptr[i].Getsubject()); strcat(querystr,"')");
       res=mysql_query(&my_connection, querystr);
       if (res) {
            cout << "Insert lev_0_list items: " << Stackptr[i].Getsubject() << " error !" <<
endl;
       }
    }
}
```

4.5.3 Keyword class

The keyword class can be used as control terms to subject association class. It represents the association between terms and subject headings. It has three friend functions, which

will be used in the subject extraction process. It also has four private data members: The Key, which holds the value of the keyword, and three pointers pointing to stack_array_0, stack_array_1 and stack_array_2 respectively. It contains redundant functions as mentioned in figure 4. The operation *isKey* checks if the string 'str' is a keyword. The

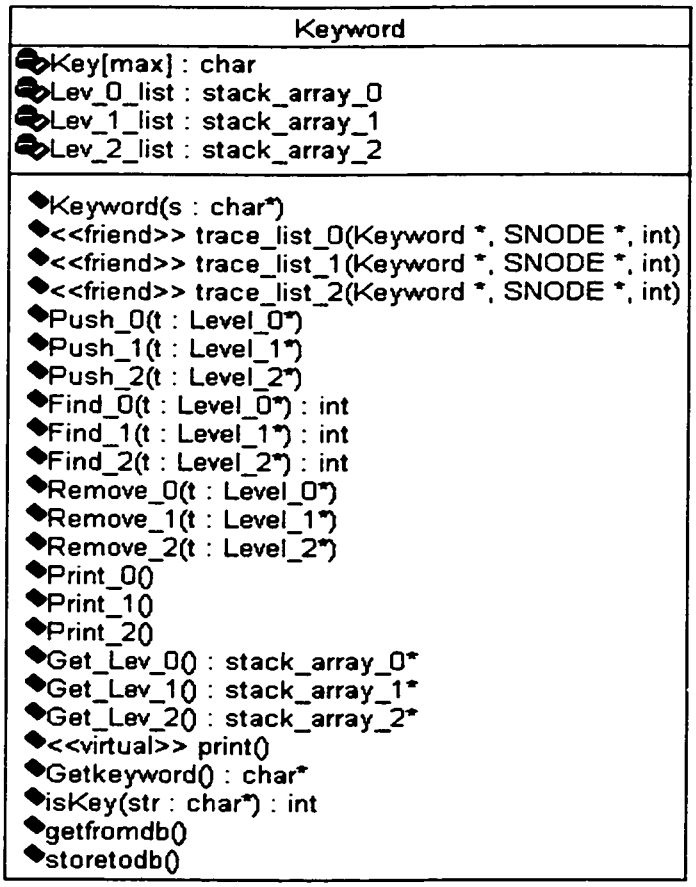

Figure 4: The class of Keyword

operation *getfromdb()* is to get the private data of the keyword class from the Thesaurus. The operation *storetodb()* is to store the private data of keyword class to the Thesaurus.

4.5.4 The table schemas of the Thesaurus

The Thesaurus database in this application is called ashg. There are in total seven tables. The following are these tables and their schemas.

Level_0

Column Name	Description
Level0	Primary key for this table
Items	Subject0 (general subject)

"*Level_0*" table keeps all information about subject0.

Level_1

Column Name	Description
Level0	Foreign key for *Level_0* table
Level1	Key of this table
Items	Subject1 (is derived from subject0)

"*Level_1*" table stores subject1 and the relation between subject0 and subject1. Level0 and Level1 are primary key, which represents the items is the subject of the general subject.

Level_2

Column Name	Description
Level1	Foreign key for *Level_1* table
Level2	Key of this table
Items	Subject2 (is derived from subject1)

"*Level_2*" table stores all information for subject2. Level1 and Level2 are primary key, which represents the items is the subject of the *Level_1* subject.

key_class

Column Name	Description
Idkey	Primary key of this table
Keyword	Keyword (Control Term)

"*kay_class*" table keeps all keywords

lev_0_list

Column Name	Description
Idkey	Foreign key for *key_calss* table
Id0	Key of this table
Items	Subjects (subject0)

"*lev_0_list*" table keeps the controlled term subject association between control terms and subject0.

lev_1_list

Column Name	Description
Idkey	Foreign key for *key_calss* table
Id1	Key of this table
Items	Subjects (subject1)

"*lev_1_list*" table stores the controlled term subject association between control terms and subject1.

lev_2_list

Column Name	Description
Idkey	Foreign key for *key_calss* table
Id2	Key of this table
Items	Subjects (subject2)

"*lev_2_list*" table is used to store the controlled term subject association between control terms and subject2.

4.5.5 Physical implementation of the tables

Here are the commands in MySQL to be used to create the tables:

```
create table level_0 (
    level0  int unsigned not null auto_increment primary key,
    items   varchar(200));

create table level_1 (
    level0  int unsigned not null,
    level1  int unsigned not null auto_increment,
```

```
    items    varchar(200),
    primary key(level0, level1),
    foreign key(level0) references level_0(level0)
);

create table level_2 (
    level1   int unsigned not null,
    level2   int unsigned not null auto_increment,
    items    varchar(200),
    primary key(level1, level2),
    foreign key(level1) references level_1(level1)
);

create table key_class (
    idkey     int unsigned not null auto_increment primary key,
    keyword   varchar(100));

create table lev_0_list (
    idkey    int unsigned not null,
    id0      int unsigned not null auto_increment,
    items    varchar(200),
    primary  key(idkey, id0),
    foreign  key(idkey) references key_class(idkey)
);

create table lev_1_list (
    idkey    int unsigned not null,
    id1      int unsigned not null auto_increment,
    items    varchar(200),
    primary  key(idkey, id1),
    foreign  key(idkey) references key_class(idkey)
);

create table lev_2_list (
    idkey    int unsigned not null,
    id2      int unsigned not null auto_increment,
    items    varchar(200),
    primary  key(idkey, id2),
    foreign  key(idkey) references key_class(idkey)
);
```

4.6 Programs used to maintain the Thesaurus

Haddad [8] has created the subject hierarchy manually for Computer Science and Electrical Engineering, which is stored in files COMPUTER_HIERARCHY and ELECTRICAL_HIERARCHY. Haddad also provide some programs to maintain the Thesaurus.

Perl script *build_sub2*: This function will generate or distribute the *Level_1* subject to *Level_2* subject headings. Applying this program on the file that contains the subject headings will generate another file having *_sub2* as an extension. This file contains the subject headings in a different format.

new-subject-build.cpp: This program can be applied on the file having *_sub2* as an extension to produce the three level subject headings hierarchy in the Thesaurus.

For example: new_subject_build COMPUTER_HIERARCHY_sub2.

Perl script *generating_added_words*: This function generates the new words not in *spell* dictionary. It passes the files that contains the subject hierarchies as inputs into some files:

1. A file called ADDED_WORDS_LIST, which contains the new added words list.
2. SUB_JECT_0_1_HEADINGS, which contains Level_0 and Level_1 subjects.
3. SUBJECT_2_HEADINGS, which contains Level_2 subjects and the keywords that are associated to them.

Perl script *stemming*: This function will take the three files mentioned above to generate a file, SUBJECT_HEADINGS_STEM_sub, which contains the root words and their corresponding subject headings. This function will call the three following functions:

Perl script *subject_words*: will divide each subject heading into its words.

Perl script *find_root*: will generate the root of the words.

Perl script *keyword_subject*: will associate the root words to their subject headings.

build_keyword_db.cpp: This program will construct the controlled term subject association by reading the files that contain a list of root controlled terms and each is followed by the subject heading where it was found.

For example: build_keyword_db SUBJECT_HEADINGS_STEM_sub.

Chapter 5

Porting ASHG

5.1 Introduction

The design goal of ASHG is to automatically build a reliable draft Semantic Header for HTML, Latex, Text and RTF documents; draft Semantic Header includes classifying a document under a list of subject headings. The ASHG provides an initial set of subject classifications and a number of components of the Semantic Header for the document. ASHG's scheme is measuring both the occurrence frequency and positional weight of keywords found in the document. Based on the selected document's keywords, the ASHG assigns a list of subject headings by matching those keywords with the controlled terms found in the controlled term subject association [8].

When a user uploads a document to the system, the ASHG extracts some fields such as document's title, abstract, keywords, dates, author, author's information, size and type from the document. Using frequency occurrence and positional schemes, the ASHG measures the significance of the words found in the previously mentioned list. Word stemming is used in order to generate a base form for each word. The system tries to match the base forms of the words with the controlled terms found in the controlled term subject association. If a match is found, the subject associated with the controlled terms are extracted and ranked accordingly.

The functions are described as follows:

1. Upload the document: The users submit a document to the system.
2. Recognizing Document Type: Currently, ASHG can process HTML, Latex, Text and RTF documents.
3. File Type Validation: The user validates the file type as determined by ASHG.
4. Applying ASHG's Extractor: The extraction program corresponding to the type of document is applied to the input document.
5. Classifying ASHG's Document: The corresponding subject headings are assigned to the document.
6. Output Semantic Header: The generated Semantic Headers are parsed and stored to the temporary database tables.
7. Validating Semantic Header: The Semantic Headers generated by ASHG is presented to the user for validation and/or modification.

5.2 Document Type Recognition

After a document is uploaded to the system, the system tries to recognize the type of the document. The file naming convention is used by the system to assist in recognition of document type. If this does not work, the system will then examine the contents. If the document type remains unrecognized, the user is informed by the system, and is asked to either choose a document type or generate the Semantic Header manually.

After successful uploading, the document is passed to the Perl script FILE_TYPE_RECOGNITION. This file was ported from the Motif-based ASHG. Inside

FILE_TYPE_RECOGNITION will call a function *byname*. This function checks the document's name extension. If the extension of the file indicates that it is an HTML,

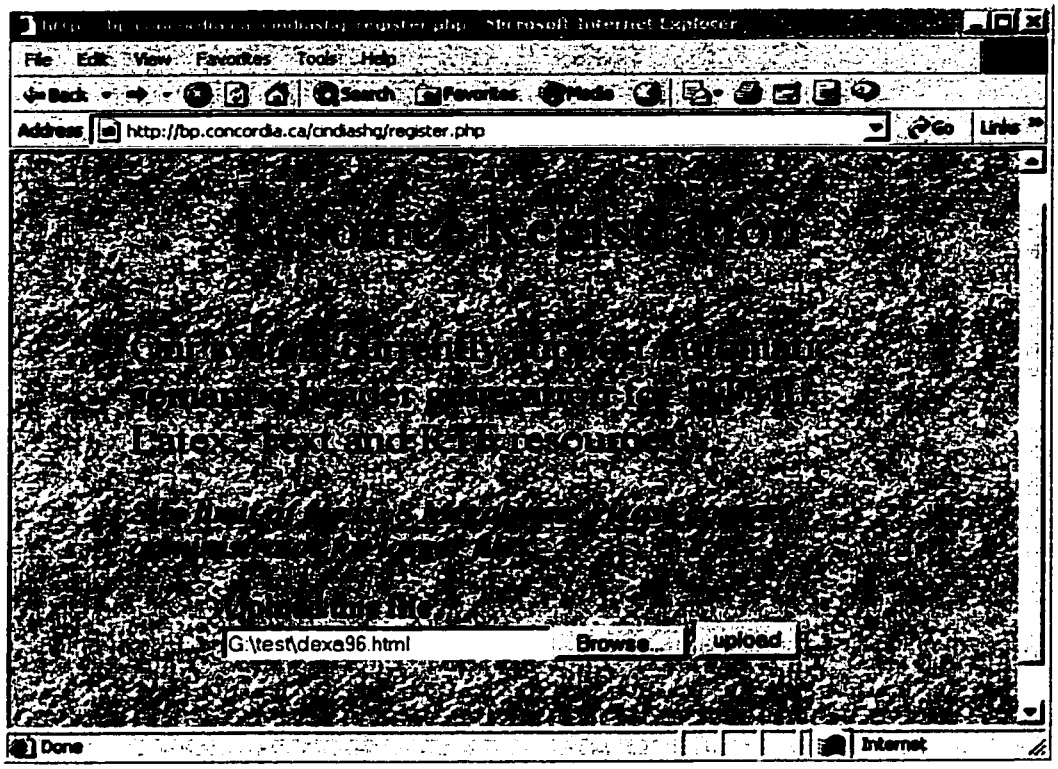

Figure 5: Uploading page

Latex, Text or RTF then the function *user_verify* is called. If the naming convention fails in recognizing the document type, the function *bycontent* is called.

```
if (document.extension == .html or .HTML or .htm) then
{
  The file is an html file.
  Call function user_verify.
}
else if (document.extension == .tex or .TEX) then
{
  The file is a latex file.
  Call function user_verify.
}
else if (document.extension == .rtf or .RTF) then
{
  The file is a rtf file.
  Call function user_verify.
}
```

```
else if (document.extension == .doc, or .txt
      or .info or .ascii) then
{
  The file is a text file
  Call function user_verify.
}
else {
  Examine the document contents by calling the function bycontent.
}
```

In the function *bycontent*, the semantics of the HTML, Latex and RTF content is exploited when attempting to recognize the file type.

```
If (the file contents match the html file semantics, such as the
  existence of the <HTML> tag) then
{
  the file is of html type
  call function user_verify
 }
else if (the file contents match the latex semantics, such as the
  existence of the \begin{...} tag) then
{
  the file is of latex type
  call function user_verify
}
else if (the file contents match the rtf semantics, such as the
  existence of the \rtf tag) then
{
  the file is of rtf type
  call function user_verify
}
else
{
  Unrecognized file type, the user should select a type.
}
```

If the file type is not HTML, Latex, Text, or RTF, it remains unrecognized, and ASHG extracts the size of the file and the date of creation.

In *user_verify*, the system redirects to a confirm web page. The user either confirms or rejects the result of this page. If the user rejects the result, he should choose a type from a list that is displayed. If the user confirms the document's type as recognized, ASHG

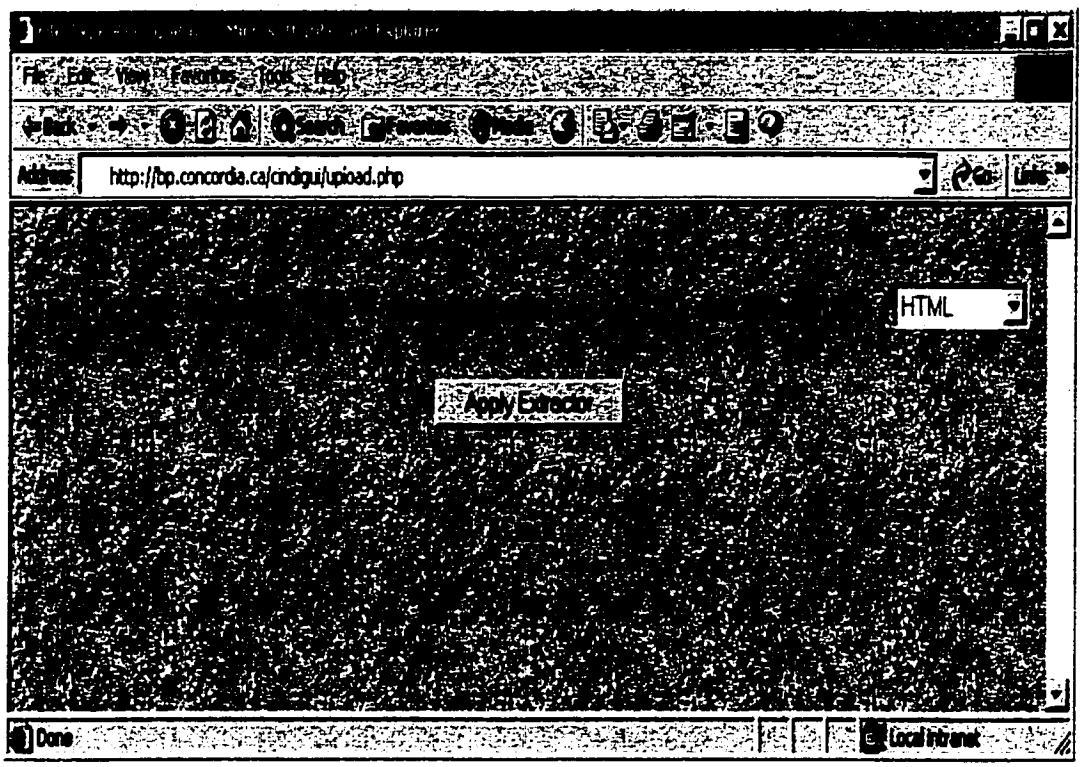

Figure 6: File type confirmation page

applies the extractor corresponding to the type confirmed by the user. Otherwise, he should choose a type and then apply ASHG.

5.3 Applying ASHG's Extractors

After the document type recognition step, ASHG applies an extraction procedure. ASHG uses its understanding of HTML, Latex, text and RTF syntax documents to extract the document's meta-information. There are five kinds of extracting programs in the system. They are HTML_extractor, LATEX_extractor, TEXT_extractor, RTF_extractor and UNKNOWN_extractor. These programs are written in Perl. They create and delete a number of temporary files while running. The original version of the extracting program

also has the function of reading the directory where there is no permission for the web user. We modified each extracting program to run on the web environment.

The other problem concerns stemming process. The original stemming process used the *spell* Sun Solaris command to extract the root of a word. The *spell* Sun Solaris command can output all the plausible derivations from the words in the spelling list. But the *spell* Linux command is different from Sun Solaris. There is no -x option to display every plausible stem in Linux. We used Porter's stemming algorithm instead of applying the *spell* Sun Solaris command with the -x option. Porter's stemming algorithm removes suffix of words to generate the root. Applying *spell* Sun Solaris command with –x option removes both prefix and suffix of words to output all available plausible stems. The original stemming process [8] was modified in the web-based ASHG.

The steps of the original stemming process are:

1. Using the *sort* command, sort the input words.
2. Apply the *uniq* command filter out duplicate words.
3. Apply the *spell* command with –x option. Thus, all the plausible stems are stored in an output file.
4. Apply the *spell* command with –v option. All words not found in the spelling list are in an output file.
5. Create a file, which contains the words found in step 3 and step 4.

The steps of the web-based ASHG stemming process are:

1. Using the *sort* command, sort the input words.
2. Apply the *uniq* command filter out duplicate words.
3. Apply the Porter's stemming algorithm

4. Apply the *spell* command. All words not found in the spelling list are in an output file.
5. Apply Perl script *checker* to check if the words not found in the spelling list are in the file ADDED_WORDS_LIST.
6. Create a file, which contains the words found in step 4 and step 5.

The file ADDED_WORDS_LIST stores all the new words not found in system's dictionary.

5.4 ASHG's Document Subject Headings Classification scheme

An important step in constructing the semantic header is to automatically assign subject headings to the documents. Sub_ext.cpp is an important program for classifying information resource. The corresponding subject headings are assigned to the document after running sub_ext. It was rewritten using C++ and MySQL to replace O++ and ODE, the interfaces between C++, MySQL and O++, ODE are different, and its algorithm was kept.

To assign the subject headings, the ASHG uses the resulting list of significant words generated from the previous extraction programs and CINDI's controlled term subject association. The subject heading classification scheme relies on passing weights from the significant terms to their associated subjects, and selecting the highest weighted subject headings [8].

The Algorithm

Having the keywords, title words, abstract words and other tagged words, will help us select the most appropriate subjects for a given document. The following algorithm is used.

1. Three lists of subject headings are to be constructed. The list of *Level_0* subject headings, the list of *Level_1* subject headings and the list of *Level_2* subject headings.

2. For each term found in both CINDI's controlled terms and the generated list of words, the system traces the controlled term's attached list of subject headings (*list of level_0*, *level_1* and *level_2*), and adds the subject headings to their corresponding list of possible subject headings.

3. Weights are also assigned to the subject hierarchies. The weight for a subject is given according to where the term matching its controlled term was found. A subject heading having a term or set of terms occurring in both title and abstract, for instance, gets a weight of seven. The matched terms' weights are passed to their subject headings.

4. The system extracts *Level_2*, *Level_1* and *Level_0* subject headings having the highest weights from the three lists of possible subject headings.

5. After building the three lists for the three level subject headings, the system:

    1) Selects the subjects using the bottom-up scheme.

    2) Having selected the highest weighted *level_2* subject headings, the system derives their *level_1* parent subject headings.

    3) An intersection is made between the derived *level_1* subject headings and the list of the highest weighted *level_1* subject headings. The common *level_1* subjects are the document's *level_1* subject headings.

    4) The system uses the same procedure in selecting *level_0* subject headings.

5.5 Semantic Header Validation

Once the process of extracting the meta-information is completed, the ASHG displays to a web page with the generated semantic header is displayed for the information resource provider to modify, add or remove some of the attributes. An example is given in figure 7

and 8. Once the provider finishes, the semantic header can be stored in the CINDI database.

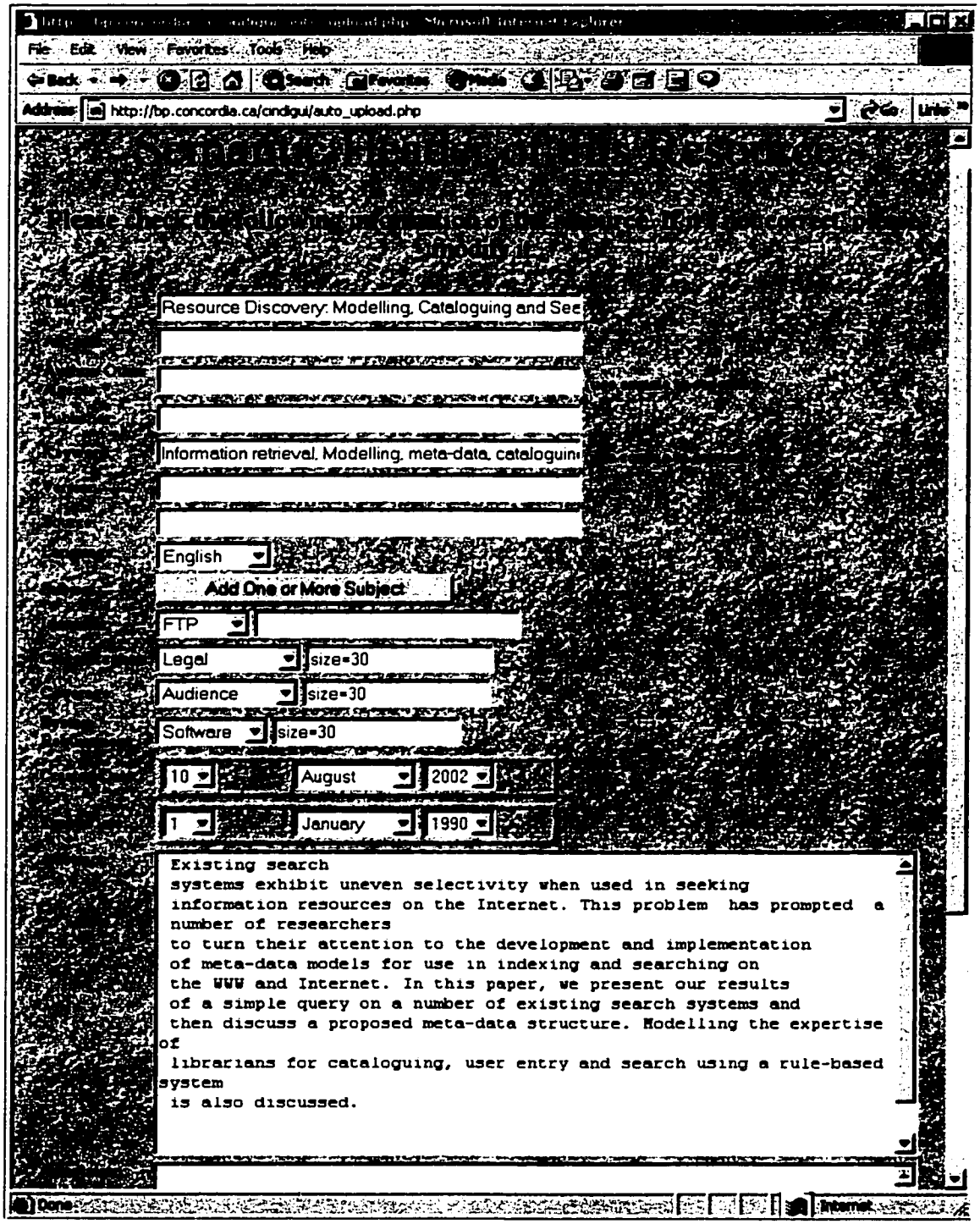

Figure 7: Semantic Header Validation

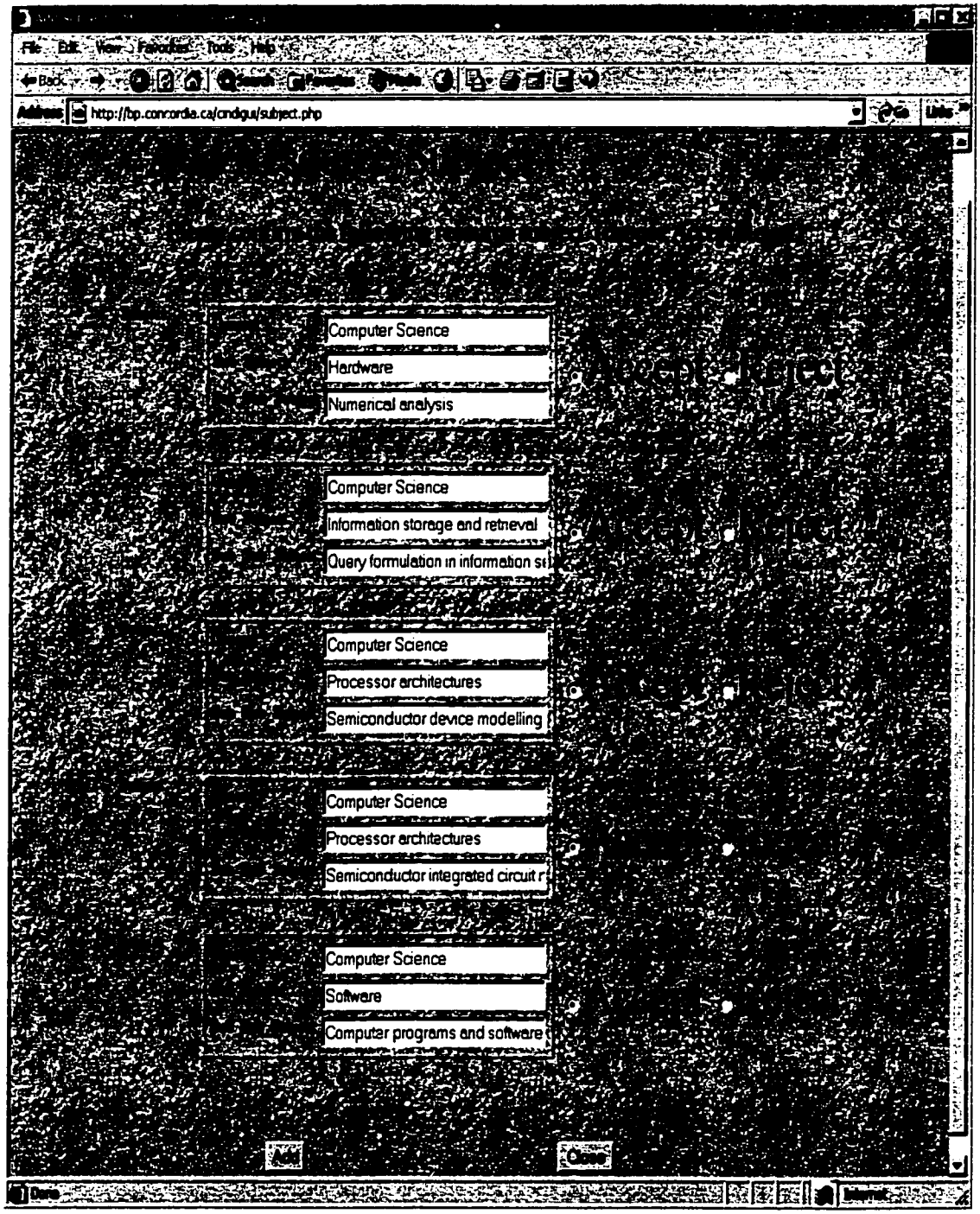

Figure 8: Subject Heading Validation

Chapter 6

Integrating and Testing

6.1 Integrating

The ASHG sub-system is the part of CINDI system. To integrate it with the CINDI, we need to consider some situations:

6.1.1 Multiple Users vs. File Name and Type

File upload function is a multi-user function. There is a possibility that two users are using a same file name. A more complex case is two users are uploading the files with the same name and at the same time. To avoid the file being overwritten in case, we use a synchronized mechanism to generate a number for each file. The different files with the same name are assigned different numbers. This number is unique inside web server. It will be held as a session variable when a user accesses the system.

After uploading the document, the web user interface of the ASHG will call the Perl script function to recognize document type. Multiple users can call this function at the same time through the web. How does it return the result of document type to the web page? We store the document type to a temporary database table. When the Perl function terminates, the web page will query the document type from the temporary table by the name of the document, which is unique and is held in the session.

Here is the implementation of the temporary table in MySQL. It has two columns.

```
create table filetype (
    filename varchar(255) NOT NULL,
    type    varchar(10)  NOT NULL,
primary key(filename));
```

6.1.2 Intermediate tables

After applying extraction program on the information resource, ASHG will generate all the Semantic Headers information to describe the document. This result will be stored in the file having same name with the document and _semantic_header as an extension.

To integrate ASHG with the CINDI system, we need some intermediate tables between ASHG subsystem and resource registration subsystem [16]. These tables are used to temporarily store Semantic Header information that is automatically generated by ASHG. The Semantic Header information generated by the ASHG is only a draft. The user needs to check and confirm it. This information cannot be directly stored in the Semantic Header database, because this will give the database false information and render the Semantic Header database useless. The main information, such as title, keyword, subject, created_date and abstract, etc., is stored in the table *sh*. The author's information of the resource is stored in the *sh_author* table. All available subject headings are stored in *sh_subject*. Information about coverage, identifier, system requirements and classification is stored in the *sh_covearage* table, the *sh_indentifier* table, the *sh_system_req* table and the *sh_classification* table correspondingly. The implementation of these tables is given in reference [16].

We give C program, called readsh.c, to store Semantic Headers to these intermediate tables. The temporary semantic header tables are to integrate the ASHG with the CINDI, readsh.c also. This function reads a Semantic Header file, parses the Semantic Header's

schema and stores the information in these temporary Semantic Header tables, and is called after the Semantic Header file to be generated by the ASHG. In Motif-based ASHG there is a similar function, which parses the Semantic Header to display it on user interface.

6.1.3 Security

For a web application, there are two kinds of security. One is system level security and another is application level security. The system level security is more concerned with locking server resource and only allowing permitted users to access. The application level security means that the application program controls the security. For example, the application uses user name and password for the system. The CINDI system uses application level security. The ASHG uses system level security since ASHG always run at the back end. We give read and execute permissions to other users for all Perl script programs, and give execute permission to other users for all C/C++ executable programs. For the directory where we store the uploading documents, we give read and write permissions. There are five extraction programs in ASHG. They create a lot temporary files when they are running. So we give the directory where the ASHG web default is read and write permission for other users. This is a weaken security. We tried to use wrappers and sudo, but failed, it does not work. Maybe we did wrong way.

In this application, there are a lot function calls using "system" command between C/C++ and Perl, PHP and Perl, PHP and C/C++. We use the absolute path to call the executable program since it is not easy to give relative path to the web default directory.

6.2 Test Results

We apply the ASHG on a set of four type documents; the titles of these documents can be viewed in the appendix. The generated index fields such as title, keywords, abstract and author are compared with those that are found in the document.

The experiments were conducted on eighteen documents to test the accuracy of the generated index and the subject heading classification results. The ASHG was able to extract all the explicitly stated fields such as title, abstract, keywords and author's information with a hundred percent accuracy. If the abstract was not explicitly stated, the ASHG was able to automatically generate an abstract that would describe the paper. However, ASHG's implicit keyword extraction generated a list of words that included some words that are insignificant. These insignificant words in turn lead to the diversion in subject classification [8].

We have tested eighteen documents in two platforms: web-based ASHG and Motif based ASHG. The ASHG's automatic subject heading classification results are compared in detail. We show the generated subject headings in four tables.

We have compared the results for HTML_extractor for HTML documents, Latex_extractor for Latex documnets, Text_extractor for text documents and RTF_extractor for RTF documents with the result for the same documents in original Motif ASHG.

The results showed two systems are same in extracting the explicitly stated fields such as the title, abstract, author and keywords. And hence we have not included these. We only show the result of subject headings for the web-based ASHG and Motif-based ASHG.

The difference was because of the stemming process, since we changed the stemming process. The stemming process is also used to create the controlled term association, so we need to create this association manually again.

We believe that the Motif-based ASHG is more accurate. The stemming is important for the performance of the system. It is better stemming process to apply *spell* Sun Solaris command with -x option. Stemming process affects extracting key words from information resource and classifying the document. It is too complicated to give some specific examples in this report and we will discuss the reason in 6.3 the analysis of testing results.

The response of the web-based ASHG is better then Motif-based ASHG, but it is not fast enough. To improve the response, we can remove all displaying information for debugging, and it is a good idea using RAM disk and reading or writing the files to it instead of hard disk.

For uploading file, there is a limit on the size of file transferred in the web-based ASHG. The limit came from the configuration of PHP script engine. Its default value is two Meg to allow user upload file, but it can be set to a different value in the PHP.ini file.

HTML Document	Number of Subject Headings generated by Web-based ASHG	Number of Subject Headings generated by Motif-based ASHG	Number of same Subject Headings	Number of different Subject Headings
D1	9	6	5	5
D2	4	9	3	7
D3	5	6	4	3
D4	1	4	1	3
D5	3	6	3	3
D6	5	6	4	3
D7	4	3	2	3
D8	3	5	2	4
D9	19	5	4	16
D10	5	7	5	2
D11	8	4	2	8
D12	5	5	2	6
D13	8	5	2	8
D14	7	4	3	4
D15	8	26	2	30
D16	4	26	3	24
D17	3	9	1	10
D18	4	3	2	3

Table 1:HTML test results

LATEX Document	Number of Subject Headings generated by Web-based ASHG	Number of Subject Headings generated by Motif-based ASHG	Number of same Subject Headings	Number of different Subject Headings
D1	5	5	5	0
D2	4	11	3	9
D3	4	6	4	2
D4	1	4	1	3
D5	3	6	2	5
D6	7	5	5	2
D7	4	5	4	1
D8	3	5	1	6
D9	4	6	3	4
D10	5	5	5	0
D11	3	3	3	0
D12	7	2	1	7
D13	7	4	3	5
D14	9	7	7	2
D15	5	24	2	25
D16	4	25	2	25
D17	3	5	2	4
D18	4	4	2	4

Table 2:Latex test results

TEXT Document	Number of Subject Headings generated by Web-based ASHG	Number of Subject Headings generated by Motif-based ASHG	Number of same Subject Headings	Number of different Subject Headings
D1	5	5	5	0
D2	26	12	2	26
D3	5	6	5	1
D4	4	10	2	10
D5	1	3	0	4
D6	16	7	2	18
D7	4	4	1	6
D8	6	4	4	2
D9	6	3	1	7
D10	3	9	0	12
D11	5	7	2	8
D12	7	31	2	34
D13	7	27	3	28
D14	5	3	2	4
D15	3	4	2	3
D16	5	6	2	7
D17	3	3	2	2
D18	8	14	4	14

Table 3:Text test results

RTF Document	Number of Subject Headings generated by Web-based ASHG	Number of Subject Headings generated by Motif-based ASHG	Number of same Subject Headings	Number of different Subject Headings
D1	7	5	5	2
D2	6	8	5	4
D3	4	5	4	1
D4	1	3	1	2
D5	12	3	3	9
D6	3	5	2	4
D7	3	4	2	3
D8	11	6	6	5
D9	5	5	4	2
D10	9	5	5	2
D11	4	3	2	2
D12	6	4	1	7
D13	2	4	2	0
D14	7	8	6	3
D15	7	8	1	13
D16	4	15	2	15
D17	3	3	1	4
D18	4	5	2	5

Table 4:RTF test results

6.3 The analysis of Test Results

It seems that the subject headings generated by the web-based ASHG are very different from the Motif-based ASHG. We can analyze this difference.

The Motif-based ASHG consists of thesaurus programs, extractors, stemming process, document type recognition, subject heading classification programs and user interface programs. We use the thesaurus programs to create and maintain ASHG Thesaurus. The extractors are used to extract Semantic Header from input document. It will use stemming process to find the root of the word extracted from document. The subject heading classification can be used to generate subject headings according on ASHG's Thesaurus and the keywords extracted by the extractors.

In web-based ASHG, the document type recognition program is the same as the Motif-based ASHG's. The extractors are modified. The thesaurus programs, stemming process, subject heading classification programs and user interface are rewritten.

We only made one change in the extractors for security reason. The function of listing all contents in one directory, which is not allowed in web environment, is changed to search for the four files named ABSTRACT_WORDS, KEYWORDS_WORDS, OTHER_WORDS, and TITLE_WORDS from the directory. So the extractors are exactly the same as in the Motif-based ASHG.

We find that there is one bug in Motif-based ASHG's thesaurus programs. The bug is that there is a trailing space in subject headings of the Motif-based ASHG and no trim function in thesaurus programs; due to this extra space, the Motif-based ASHG's thesaurus is not correct. For example, "science" is keyword and has an association with

"computer science" in *level_0*. Actually, "computer science_", where _ is used to indicate a space, is stored in Motif-based ASHG's thesaurus instead of "computer science". So this association is lost. The "use" is also a key word and associates with "user interfaces", but there are two associations stored in Motif-based ASHG's thesaurus. One is "user interfaces". The other one is "user interfaces_". There are lots of subject headings and keywords include space at the end of them in Motif-based ASHG's thesaurus. This means that controlled term associations is not correct in old system. In the Motif-based ASHG, we fixed this bug for the version made by Ali [7].

We tested and debugged the web-based ASHG module by module and compared with Modif-based ASHG, and found some API functions of MySQL automatically remove trailing space from string, but the trailing space was kept in O++ and ODE.

We have ensured that the functions of subject heading classification program (sub_ext) in web-based ASHG are the same as in Motif-based ASHG. We use the same input file (WORDS_TO_SUBJECT) containing the available keywords and its weight to test subject heading classification programs in web-based and Motif-based ASHG. They generate the same subject headings.

Here is a sample for one document:

The file WORDS_TO_SUBJECT generated by the Motif-based ASHG's extractor contains the available keywords and its weight as follows:

```
call 2
client 6
computation 6
constraint 2
database 6
efficient 2
extend 2
implement 2
involve 6
limit 6
```

linear 2
numeric 6
package 6
paper 2
pay 6
problem 6
product 2
project 2
query 2
select 2
send 6
solution 2
table 6
time 5
use 2
weight 6

After running sub_ext in web-based and Motif-based ASHG, the same result was generated for both sub_exts as follows:

option 1:
Computer Science
 Software
  Computer programs and softwares
---
option 2:
Computer Science
 Analysis of algorithms and problem complexity
  Geometrical problems and nonnumerical computations
---
option 3:
Computer Science
 Programming languages
  Computer program language
---
option 4:
Computer Science
 Analysis of algorithms and problem complexity
  Numerical computation of transforms
---
option 5:
Computer Science
 Analysis of algorithms and problem complexity
  Fast fourier numerical computation of transforms
---
option 6:
Computer Science
 Analysis of algorithms and problem complexity
  Numerical computations in finite fields
---
option 7:
Computer Science
 Analysis of algorithms and problem complexity

Numerical computations on matrices
---
option 8:
Computer Science
Analysis of algorithms and problem complexity
Numerical computations on polynomials
---
option 9:
Computer Science
Analysis of algorithms and problem complexity
Number-theoretic numerical computations
---
option 10:
Computer Science
Analysis of algorithms and problem complexity
Number-theoretic numerical computations: factoring
---
option 11:
Computer Science
Analysis of algorithms and problem complexity
Number-theoretic numerical computations: primality testing
---
option 12:
Computer Science
Analysis of algorithms and problem complexity
Computations on nonnumerical discrete structures
---
option 13:
Computer Science
Analysis of algorithms and problem complexity
Numerical algorithms and problems
---
option 14:
Computer Science
Analysis of algorithms and problem complexity
Nonnumerical algorithms and problems

We did this test for eighteen documents. Both sub_ext generated the same results.

The stemming process is different in the two systems. Stemming process in web-based ASHG, Porter's stemming algorithm, removes suffix of words to generate the root. Motif-based ASHG stemming, applying *spell* Sun Solaris command with –x option, removes both prefix and suffix of words to output all available plausible stems. Changing stemming process will affect the result file of the extractors. It also affects all available keywords stored in Thesaurus, which are extracted from subject headings using corresponding stemming process.

The reasons mentioned above make the subject headings generated by two systems different, as shown in the sample results below.

6.4 Sample Results

In this section, we will show some of indexes generated for document D1 by two systems.

Sample 1: generated by the Motif-based ASHG.

<semhdrB>
<useridB> <useridE>
<passwordB> <passwordE>
<titleB> Resource Discovery: Modelling, Cataloguing and Searching <titleE>
<alttitleB> <alttitleE>
<subjectB>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> Information search and retrieval <sublevel2E>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> Query formulation in information search and retrieval <sublevel2E>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> Relevance feedback in information search and retrieval <sublevel2E>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> Retrieval models in information search and retrieval <sublevel2E>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> Information search and retrieval process <sublevel2E>
<subjectE>
<languageB> English <languageE>
<char-setB> <char-setE>
<authorB>
<aroleB> Author <aroleE>
<anameB> Bipin C. DESAI, ~~ ~~Rajjan SHINGHAL <anameE>
<aorgB> <aorgE>
<aaddressB> Department of Computer Science, Concordia University, Montreal, H3G 1M8, CANADA
<aaddressE>
<aphoneB> <aphoneE>
<afaxB> <afaxE>
<aemailB> <aemailE>
<authorE>
<keywordB> Information retrieval , Modelling , meta-data , cataloguing searching , discovery , information resources , WWW , Internet , resource discovery <keywordE>
<identifierB>

<domain3B> FTP <domain3E>
<value3B> <value3E>
<identifierE>
<datesB>
<createdB> 2001/7/3 <createdE>
<expiryB> <expiryE>
<datesE>
<versionB> <versionE>
<spversionB> <spversionE>
<classificationB>
<domain4B> <domain4E>
<value4B> <value4E>
<classificationE>
<coverageB>
<domain5B> <domain5E>
<value5B> <value5E>
<coverageE>
<system-requirementsB>
<componentB> <componentE>
<exiganceB> <exiganceE>
<system-requirementsE>
<genreB>
<formB> <formE>
<sizeB> 42301 <sizeE>
<genreE>
<source-referenceB>
<relationB> <relationE>
<domain-identifierB> <domain-identifierE>
<source-referenceE>
<costB> <costE>
<abstractB>
Existing search
systems exhibit uneven selectivity when used in seeking
information resources on the Internet. This problem has prompted a
number of researchers
to turn their attention to the development and implementation
of meta-data models for use in indexing and searching on
the WWW and Internet. In this paper, we present our re-sults
of a simple query on a number of existing search systems and
then discuss a pro-posed meta-data structure. Modelling the expertise of
librarians for cataloguing, user entry and search using a rule-based system
is also discussed.
<abstractE>
<annotationB>
<annotationE>
<semhdrE>
<EOF>

Sample 2: generated by the web-based ASHG.

<semhdrB>
<useridB> <useridE>
<passwordB> <passwordE>
<titleB> Resource Discovery: Modelling, Cataloguing and Searching <titleE>
<alttitleB> <alttitleE>

<subjectB>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> information search and retrieval <sublevel2E>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> query formulation in information search and retrieval <sublevel2E>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> relevance feedback in information search and retrieval <sublevel2E>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> retrieval models in information search and retrieval <sublevel2E>
<generalB> Computer Science <generalE>
<sublevel1B> Information storage and retrieval <sublevel1E>
<sublevel2B> information search and retrieval process <sublevel2E>
<subjectE>
<languageB> English <languageE>
<char-setB> <char-setE>
<authorB>
<aroleB> Author <aroleE><anameB> Bipin C. DESAI, ~~ ~~Rajjan SHINGHAL <anameE>
<aorgB> <aorgE>
<aaddressB> Department of Computer Science, Concordia University, Montreal, H3G 1M8, CANADA
<aaddressE>
<aphoneB> <aphoneE>
<afaxB> <afaxE>
<aemailB> <aemailE>
<authorE>
<keywordB> Information retrieval , Modelling , meta-data , cataloguing searching , discovery , information resources , WWW , Internet , resource discovery <keywordE>
<identifierB>
<domain3B> FTP <domain3E>
<value3B> <value3E>
<identifierE>
<datesB>
<createdB> 2002/9/16 <createdE>
<expiryB> <expiryE>
<datesE>
<versionB> <versionE>
<spversionB> <spversionE>
<classificationB>
<domain4B> <domain4E>
<value4B> <value4E>
<classificationE>
<coverageB>
<domain5B> <domain5E>
<value5B> <value5E>
<coverageE>
<system-requirementsB>
<componentB> <componentE>
<exiganceB> <exiganceE>
<system-requirementsE>
<genreB>
<formB> <formE>
<sizeB> 42301 <sizeE>
<genreE>

<source-referenceB>
<relationB> <relationE>
<domain-identifierB> <domain-identifierE>
<source-referenceE>
<costB> <costE>
<abstractB>
Existing search
systems exhibit uneven selectivity when used in seeking
information resources on the Internet. This problem has prompted a
number of researchers
to turn their attention to the development and implementation
of meta-data models for use in indexing and searching on
the WWW and Internet. In this paper. we present our re-sults
of a simple query on a number of existing search systems and
then discuss a pro-posed meta-data structure. Modelling the expertise of
librarians for cataloguing, user entry and search using a rule-based system
is also discussed.
<abstractE>
<annotationB>
<annotationE>
<semhdrE>
<EOF>

Chapter 7

Conclusion and Future Work

7.1 Conclusions

The Internet is quickly evolving from today's Web sites that just deliver user interface pages to browsers to a next generation of programmable Web sites that directly link organizations, applications, services, and devices with one another. There has been much interest recently in moving data from the WWW into database. The ASHG is a useful application, a kind of service. We presented the porting a software package of generating a Semantic Header to the web. The web-based ASHG presents a way of porting the application in Motif, TCP/IP based system to web environment. It implemented a totally free, robust and secure web service.

We have integrated the ASHG sub-system with the CINDI system. The result shows that PHP-MySQL-Apache-Linux is a good, robust and secure combination for porting an application to web.

In this project, the Thesaurus database was redesigned and implemented in the web server. The subject headings extract module based on MySQL was implemented. Data manipulation software for the Thesaurus was provided. We changed the stemming process; and a controlled term subject heading association was improved. Four extraction programs; HTML_extractor for HTML documents, Latex_extractor for Latex documnets, Text_extractor for text documents and RTF_extractor for RTF documents were integrated

with the ASHG. Web-based user interface was implemented and easily integrated with the CINDI. A new ability of uploading document was added.

Lastly, we applied ASHG to a collection of four types of testing documents and compared the results to the Motif-based ASHG in Unix.

7.2 Future Work

In the future work, some new functions could be built and some existing functions could be improved.

The semantic level language processing should be handled by ASHG. Since ASHG was only based on the frequency and location of words in a document to determine the document's keywords and subject headings and subject classification, it has missed the importance of the word senses and the relationship between words in a sentence.

More subject hierarchies, such as Civil Engineering, and Mechanical Engineering should be built.

The extractor for PDF document to improve the performance could be added in the ASHG to understand PDF type document. The ASHG can currently handle PDF type document by converting it into text type documents.

For security reasons, the four extraction programs will be modified. They will not be able to create temporary files in web default directory during their running. If they have to store temporary data, a database can be used. In this way, we can only grant read permission to web default directory for common users.

Besides, we may separate the database server from the web server now that we only use one server to act as both web server and DB server. It is always good to set a DB server standalone.

Appendix

Papers Used in Testing ASHG

The following is the list of papers used in testing ASHG for RTF_extractor

D1 Desai B. C., and Shinghal R., *Resource Discovery: Modelling, Cataloguing and Searching*, Department of Computer Science, Concordia University, Montreal, Canada.

D2 Grogono P., *Designing for Change*, Department of Computer Science, Concordia University, Montreal, Canada.

D3 Grogono P., *Designing a class library*, Department of Computer Science, Concordia University, Montreal, Canada.

D4 Grogono P., *A Code Generator for Dee*, Department of Computer Science, Concordia University, Montreal, Canada.

D5 Butler G., Grogono P., Shinghal R., and Tjandra I., *Retrieving Information from Data Flow Diagrams*, Department of Computer Science, Concordia University, Montreal, Canada.

D6 Grogono P. and Santas P., *Equality in Object Oriented Languages*, Department of Computer Science, Concordia University, Montreal, Canada and Institute of Scientific Computation, ETH Zurich, Switzerland.

D7 Grogono P. and Gargul M., *A Computational Model for Object Oriented Programming*, Department of Computer Science, Concordia University, Montreal, Canada.

D8 Grogono P. and Gargul M., *A Graph Model for Object Oriented Programming*, Department of Computer Science, Concordia University, Montreal, Canada.

D9 Grogono P. and Gargul M., *Graph Semantics for Object Oriented Programming*, Department of Computer Science, Concordia University, Montreal, Canada.

D10 Grogono P., *Issues in the Design of an Object Oriented Programming Language*, Department of Computer Science, Concordia University, Montreal, Canada.

D11 Paknys R. and Raschkowan L. R., *Moment Method Surface Patch and Wire Grid Accuracy in the Computation of Near Fields*, Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada.

D12 Paknys R., *On the Accuracy of the UTD for the Scattering by a Cylinder*, Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada.

D13 Davis D., Paknys R., and Kubina S. J., *The Basic Scattering Code Viewer A GUI for the NEC Basic Scattering Code*, Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada.

D14 Ounis I., Pasca M., *An Extended Inverted File Approach for Information Retrieval*, Grenoble, France.

D15 Ludwig A., Becker P. and Guntzer U., *Interfacing Online Bibliographic Databases with Z39.50*, University of Tubingen, Germany.

D16 Cho E. S., Han S. Y., Kim H. J. and Thor M. Y., *A New Data Abstraction Layer Required For OODBMS*, Department of Computer Science and Computer Engineering, Seoul National University, Seoul, Korea.

D17 Ehikioya S. A., *A Formal Specification Strategy for Electronic Commerce*, Department of Computer Science University of Manitoba, Winnipeg, Manitoba, Canada.

D18 Revesz P. Z. and Li Y., *MLPQ: A Linear Constraint Database System with Aggregate Operators*, Dept. of Computer Science and Engineering, Lincoln, USA.

References

[1] Desai B. C., Cover page aka Semantic Header, http://www.cs.concordia.ca/~faculty/bcdesai/semantic-header.html, July 1994, revised version, August 1994.

[2] Desai B. C., The Semantic Header Indexing and Searching on the internet, February 1995. http://www.cs.concordia.ca/ faculty/bcdesai/cindi-system-1.1.html

[3] Desai, B. C., An Introduction to Database Systems, West, St. Paul, MN 1990.

[4] Edmundson H. P. and Wyllys R. E., Automatic Abstracting and Indexing Survey and Recommendations, Communications of ACM, 4:5, pp. 226-234, May 1961.

[5] Fung R. and Del Favero B. Applying Bayesian Networks to Information Retrieval, Communications of the ACM, Vol 38, No. 3, pp. 42-57, March 1995

[6] Bipin C. Desai, Shinghal Rajjan, *A System for Seamless Search of Distributed Information Sources*, May 1994. http://www.cs.concordia.ca/~faculty/bcdesai

[7] Abdelbaset Ali, Extraction of Semantic Header from RTF Document. Major Report, Department of Computer Science, Concordia University, Montreal, Canada, 1999.

[8] Haddad S., ASHG: *Automatic Semantic Header Generator.* Master's thesis, Department of Computer Science, Concordia University, Montreal, Canada, 1998.

[9] Shayan N., CINDI: *Concordia INdexing and DIscovery system.* Master's thesis, Department of Computer Science, Concordia University, Montreal, Canada, 1997.

[10] Youquan Zhou, CINDI: *The virtual Library Graphical User Interface.* Master's thesis, Department of Computer Science, Concordia University, Montreal, Canada, 1997.

[11] Kurt Wall, *Linux Programming (ISBN 0-7897-2215-1),* QUE, 1999

[12] Bill Ball, David Pitts, *Red Hat Linux 7 (ISBN 0-672-31985-3),* SAMS, 2000

[13] MySQL 3.23.29a-gamma Reference

[14] *UNIX Learning Perl* Second Edition 1997 By Randal L. Schwartz and Tom Christianse.

[15] PHP Documents, http://www.php.net/docs.php

[16]Yuhui Wang, Enhanced Web Based CINDI System. Major Report, Department of Computer Science, Concordia University, Montreal, Canada, 2002.