PUBLICATION OF BIBLIOGRAPHIES ON THE WORLD WIDE WEB

by

Ibrahim Utku Moral

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Approved:

Osman Balci, Chairman
James D. Arthur
John W. Roach

January 28, 1997
Blacksburg, Virginia

Keywords: Bibliography publishing, world wide web, information storage and retrieval, bibliography search, www software development, query manipulation, database management.

Copyright 1997, Ibrahim Utku Moral

PUBLICATION OF BIBLIOGRAPHIES

ON THE

WORLD WIDE WEB

by

Ibrahim Utku Moral

Osman Balci, Chairman

Computer Science and Applications

(ABSTRACT)

Every scientific research begins with a literature review that includes an extensive bibliographic search. Such searches are known to be difficult and time-consuming because of the vast amount of topical material existing in today's ever-changing technology base. Keeping up-to-date with related literature and being aware of the most recent publications require extensive time and effort. The need for a WWW-based software tool for collecting and providing access to this scientific body of knowledge is undeniable. The study explained herein deals with this problem by developing an efficient, advanced, easy-to-use tool, WebBiblio, that provides a globally accessible WWW environment enabling the collection and dissemination of searchable bibliographies comprised of abstracts and keywords. This thesis describes the design, structure and features of WebBiblio, and explains the ideas and approaches used in its development. The developed system is not a prototype, but a production system that exploits the capabilities of the WWW. Currently, it is used to publish three VV&T bibliographies at the WWW site: http://manta.cs.vt.edu/biblio. With its rich set of features and ergonomically engineered interface, WebBiblio brings a comprehensive solution to solving the problem of globally collecting and providing access to a diverse set of bibliographies.

ACKNOWLEDMENTS

I am deeply grateful to Dr. Osman Balci, my advisor and chairman of my committee, for the many hours he has spent on my behalf, his guidance and feedback, and his incredible enthusiasm and energy, which have been great examples for me in bringing this study to completion. I want to thank him also for the countless meals we had together. I am also indebted to the other members of my committee, Dr. James Arthur and Dr. John Roach, for their review and valuable time.

All the staff of the Department of Computer Science and the Systems Research Center have always been very supportive during my entire graduate study. In particular, I want to thank to Dr. Richard Nance, Dr. Donald Allison, Dr. Calvin Ribbens, Dr. Verna Schuetz, Sandra Birch, Tammi Johnston and Sharon Donohue. I must also thank the graduate students of the Department of Computer Science, with whom I have attended classes, worked for long hours and shared the common goal. I would like to single out Bo Begole, Markus Gröner, Olivier Marchand, Anup Mathur, Craig Struble, Michael Talbert, Lucio Tinoco, Ali Tuglu, and Levent Yilmaz for their very valuable friendship and support throughout my years in the department.

Finally, I thank my family and all my friends for their support and encouragement throughout this study. They were always with me when I needed their understanding and help, especially at the times when my study took precedence over my commitments to them because of approaching deadlines. For this I am most grateful. I dedicate this thesis to my family, and to all my friends and people who were part of my education.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 : INTRODUCTION

This chapter provides a description of the research problem, states the research objectives, and gives an overview of the thesis.

1.1 Statement of the Problem

Every scientific research begins with a literature review that includes bibliographic searches throughout the related literature all over the world. Conducting bibliographic searches is known to be a difficult and time-consuming process because of the vast amount of research conducted simultaneously and fast pacing technology. Keeping up-to-date with the entire related literature and being aware of the most recent publications require extensive time and effort. The need for a software tool that can be used to publish any type of searchable scientific bibliography with abstracts and keywords on the World Wide Web is undeniable.

Currently, tools exist to deal with this problem, but they do not provide a comprehensive solution. Obtaining just a list of references is not much useful. Accessing abstracts and keywords is critical. Being able to perform advanced queries with respect to each element of a bibliographic reference as well as the abstract and keywords is very much needed. Hence, the research described herein addresses this need by focusing on the development of such a tool which is called WebBiblio (http://manta.cs.vt.edu/biblio).

The need for publishing searchable bibliographies with abstracts and keywords was originated in Dr. Osman Balci's research project funded by the Defense Modeling and Simulation Office (DMSO). One of the objectives of the project was to publish verification

and validation (V&V) bibliographies that can be accessed by researchers in the Department of Defense, industry and academia. The DMSO project needed V&V bibliographies for the following three fields: simulation and modeling, software engineering, and knowledge-based/expert systems.

Thus the problem addressed in the thesis work described herein is two folded: (1) development of a software tool for publication of searchable bibliographies with abstracts and keywords on the World Wide Web, and (2) creation of V&V bibliography databases for simulation and modeling, software engineering, and knowledge-based/expert systems.

1.2 Statement of the Objectives

The objectives of the research described herein are presented below in no particular order:

Objective 1: WebBiblio should be a globally available tool on the WWW.

The World Wide Web (WWW) is becoming the basic communication and information retrieval source of today and also of the future. It provides information worldwide in an easy and economical way. It is the new era in communication, which is growing quickly and improving continuously. Yourdon [1996] describes the WWW as a revolution in the computer science. It is the most easily accessible global communication tool for most researchers, and unless the access rights are restricted, any information provided on the WWW becomes globally available to any WWW user.

Objective 2: WebBiblio should facilitate the publication of all possible elements of a bibliographic reference structure.

This objective is based on the consideration of development of a general-purpose tool. By fulfilling this objective, WebBiblio can be used to publish any type of scientific

bibliography. This feature increases its usefulness and allows it to meet a much wider range of needs in many disciplines.

Objective 3: WebBiblio should be well human engineered.

WebBiblio should provide a user interface that satisfies the nine usability principles for interfaces [Nielsen 1990]: simple and natural dialogue, speak the user's language, minimize the user's memory load, consistency, provide feedback, provide clearly marked exits, provide shortcuts, good error messages, and prevent errors.

Objective 4 WebBiblio should provide on-line assistance.

This objective should be fulfilled by providing an easy-to-navigate, consistent help utility explaining all components of the system in detail. The help utility should contain simple explanations with a rich set of examples, and also specific definitions to meet the various needs of users with different knowledge levels efficiently.

Objective 5: WebBiblio should provide advanced search capabilities.

WebBiblio should provide its users much more than simply downloading bibliographic references. A complete, easy-to-use query form allowing simple as well as advanced query composition capabilities should be provided.

Objective 6: WebBiblio should be verified, validated and tested as much as possible.

This objective aims to increase the software's reliability, and to improve its usability by making it robust [Sommerville 1996a]. To fulfill this objective, the functionality of all the components of the software system should be completely verified, validated and tested. First, module testing should be performed for each developed individual component. Then, the entire system should be verified, validated and tested with the integration of system

components. WebBiblio is used by a worldwide range of users, who have various backgrounds and computer skills. That increases the possibility and types of user errors. Hence, WebBiblio should be also tested with as many as possible incorrect inputs, and its appropriate responses for these inputs should be verified and validated.

Objective 7: WebBiblio should provide abstracts and keywords of the sources cited by the bibliographic references.

Providing abstract and keywords information of publications in an easily accessible way together with the bibliographic information is an extremely beneficial and time-saving feature for literature surveys. Hence, WebBiblio should also provide abstracts and keywords of the sources cited by the references in the bibliography as part of reference information. However, this information should be provided upon request, not within the list of references that match to the search query, although the user should be capable to compose queries over abstract and keywords fields like every other field of a bibliographic reference.

Objective 8: WebBiblio should respond to user queries as fast as possible.

The total response time of WebBiblio consists of two basic components: data transfer time between the user's browser and WebBiblio, and the processing time of WebBiblio for the user's query. The former depends on the user's computer system and the connection speed and is beyond the scope of the development of WebBiblio. Hence, this objective implies minimizing WebBiblio's query processing time.

Objective 9: WebBiblio should be easy to administer.

The administrative tasks of WebBiblio mostly consist of updating the bibliography databases by including new references or importing contributions, and also updating and modifying the web pages. All these administrative procedures should be as simple and high level as possible. To be simple, they should consist of few steps, and to be high level, performing none of these steps should require a deep knowledge of any system components.

Objective 10: WebBiblio should be easy to maintain.

This objective is necessary to simplify the error corrections that may need to be performed as well as for future expansion of functional capabilities.

Objective 11: WebBiblio should enable anyone to contribute to the published bibliographies.

This objective is important for increasing the quality of the published bibliographies. With user contributions, which may come from different sources worldwide, the published bibliographies can be updated with most recent publications, can be expanded with new ones, and inaccurate information can be corrected.

Objective 12: A simulation verification, validation and testing bibliography database containing at least 500 references with abstracts and keywords should be developed, and WebBiblio should be used to publish it worldwide.

This objective is essential for satisfying the need of the DMSO project as well as for verification, validation and testing (VV&T) of WebBiblio. In addition to the creation of this database, a secondary objective is to create another VV&T bibliography database for knowledge-based/expert systems. Simulation VV&T database is used as a testbed for

assessing the accuracy of WebBiblio and improving and tuning the performance (execution efficiency) of WebBiblio.

1.3 Overview of the Thesis

This thesis consists of seven chapters and an appendix. Chapter 1 provides a description of the research problem, states the research objectives, and gives an overview of the thesis. Chapter 2 introduces other bibliography publishing sites currently available on the WWW and compares their characteristics with the WebBiblio's.

Chapter 3 provides a general overview of WebBiblio's software engineering. It first specifies the requirements of the software tool defined in accordance with the research objectives. Then, it describes the hardware and software environments used in developing WebBiblio. Finally, it presents WebBiblio's design specification and its general implementation structure.

Chapter 4 presents the User's Guide. It describes all WebBiblio components in detail. Chapter 5 provides the System Administrator's Manual and explains all administrative tasks step by step. Chapter 6 presents a subjective evaluation of WebBiblio with respect to the research objectives.

Finally, Chapter 7 gives a thesis summary, provides conclusions, and makes recommendations for future research.

CHAPTER 2 : LITERATURE REVIEW

This chapter compares WebBiblio with other sites on the WWW publishing bibliographies. Among these sites, four are selected for comparison: Online Computer Library Center's FirstSearch®, Achilles' "The Collection of Computer Science Bibliographies", Jones' "Searching the HPB Bibliographies" and Kamath's "Quick-Search Database Systems Research Bibliography". These four provide best features and fall within the same class as WebBiblio's. Others are not comparable to WebBiblio due to lack of significant features and are excluded form the comparison.

Section 2.1 through 2.4 present the four bibliography publishing sites and provide some comparative evaluation. Section 2.5 summarizes the comparison of features.

2.1 Online Computer Library Center's FirstSearch®

One of the most commonly used software for bibliography search is FirstSearch® [OCLC 1996]. Briefly, FirstSearch® provides web access and telnet access to its users to perform reference search over its rich set of bibliography databases. The web accessible bibliographies are classified under more than fifty databases. For instance, INSPEC is one of these that is very well known by computer scientists.

FirstSearch®'s 'Using FirstSearch' web page introduces the system as "OCLC's FirstSearch® service is an interactive online information system that gives you vital, timely information about books, journal articles, films, computer software, and other materials in your subject area. FirstSearch is very easy to use; no training is necessary. Just follow the

instructions that appear online. The FirstSearch service is a product of OCLC Online Computer Library Center, Inc., a nonprofit membership organization serving libraries and educational institutions worldwide" [OCLC 1996].

As it can be seen from the quotation, FirstSearch® is only available for a specific set of institutions in contrast to WebBiblio, which is available to any WWW user. For example, when accessing FirstSearch® at Virginia Tech over the WWW, *vt.edu* domain name is checked, and the connection is permitted only to the users under this domain name.

The same WWW page gives the information about the search techniques and limitations of FirstSearch®. Although the exact set depends on the database that is currently being searched, FirstSearch® provides searches in author, subject, title, abstract, publisher, descriptor (keywords in WebBiblio), source (published in/as in the WebBiblio), year, type and language fields. FirstSearch®'s fields are not as rich and comprehensive as the fields WebBiblio provides. Besides these fields, WebBiblio provides searches in the fields of volume number, issue number, edition, page numbers, month, editor name, publisher place and number of pages. The design of WebBiblio's *Query Form* fields is based on all possible fields in the structure of a bibliographic reference plus abstract and keywords fields. Hence the language field is not included, which is the only extra field of FirstSearch®'s query form.

On the other hand, WebBiblio provides searches in these fields with a more friendly interface and simpler syntax rules compared to the ones of FirstSearch®.

WebBiblio's *Query Form* provides a form field for every reference item to enable the composition of complex queries in contrast to FirstSearch®'s form, where the queries for most of the fields are written in the same form field and the searched reference fields are determined with the usage of some reserved words as prefixes in the query [OCLC 1996]. FirstSearch®'s approach is not easy to use for naive users and is more error-prone. It requires use of reserved words, some extra typing, and makes the entire query less readable. An advance query with parentheses can look easily very complicated, which is not the case when using WebBiblio's *Query Form*.

FirstSearch® allows the use of the logical operations: AND, OR and NOT. Parentheses can be used to change the precedence order of the operations [OCLC 1996]. This is exactly the same functionality provided by WebBiblio when the allowed logical operators in query composition are considered. Also, both of the search engines give the flexibility to the user to restrict a search on an exact match. Again, they both provide the abstract and keywords of the selected records among the query results upon request.

To conclude, when the query composition capabilities and the user interface are considered, the WebBiblio outperforms the FirstSearch® with its extra search fields, user-friendly interface and simpler syntax. On the other hand, the FirstSearch® have some extra capabilities in viewing documents, like printing or e-mailing a selected document, which the WebBiblio does not have. FirstSearch® can also provide a list of the libraries that own the bibliographic reference sources with order forms.

2.2 Achilles' "The Collection of Computer Science Bibliographies"

Another site serving as the bibliography search engine on the WWW is "The Collection of Computer Science Bibliographies" [Achilles 1996a,b,c,d]. This collection has been created and maintained by Alf-Christian Achilles. The site provides two different types of searching service. These are simple search and advanced search respectively. The simple search uses the tool Glimpse [Manber 1996] as the search engine running behind. Manber [1996] describes the search engine as "Glimpse is a very powerful indexing and query system that allows you to search through all your files very quickly. It can be used by individuals for their personal file systems as well as by organizations for large data collections." Hence, based on the searching system, the simple search of the site is limited with the string search among the files containing the bibliography database. Consequently, the query form has only one field and any query related with any field of the reference has to be written in this form field.

The advanced search section of the site has a closer structure to WebBiblio's *Query Form*. As stated on the web page of the environment titled "Help on the Computer Science Bibliography Collection Advanced Search Facility"[Achilles 1996b], the advanced search form uses the search engine freeWAIS [Pfeifer 1996b] in the background. The site provides form fields for five reference fields: type of the publication, author name, title, journal or conference name and year. These are a subset of the form fields provided by WebBiblio. However, a sixth form field in Achilles' site partially fills the gap by providing a search on any field of the bibliographic reference, although it still lacks search queries specifically and separately on every reference field.

Similar to WebBiblio’s, it supports string search as well as exact match search; but because of execution efficiency issues the usage of some strings is not allowed. This is a limitation of the site in comparison with WebBiblio. The search engine supports the usage of logical operators AND, OR and NOT and parentheses in query composition [Achilles 1996b]. This feature is the same as WebBiblio’s.

The form has only one numerical field, year, which can be compared with the numerical fields of WebBiblio's *Query Form*. This field allows only one year entry and the year entries as ranges are not fully supported; the user can make the entered year an upper bound or lower bound, but cannot specify a range with both of the boundary values since only one year can be entered in the field. The year entries are restricted to be four digits. By allowing year ranges as query as well as the individual years and also multiple queries combined with logical OR-operation, and not restricting the user to use only four digits, WebBiblio provides a much wider functionality and flexibility to its users for a numerical field. Besides, it contains numerical fields for all number containing reference fields.

The two engines have significant differences in displaying the search results. Based on the searching technique freeWAIS [Pfeifer 1996b], which is running behind the Achilles's query form, the search results can be sorted with respect to their computed match scores [Achilles 1996b], which indicate the quality of the matches [Pfeifer 1996a]. This can be an advantage over WebBiblio's approach, since WebBiblio’s search engine does not assign any match score to the matching reference. All the matching references have the same match quality value and they are displayed according to the alphabetical order of the author names.

On the other hand, when the search results are displayed, the Achilles' site supports automatic query composition based on the author name when clicked on the author name. In the contrary, the clickable author names mean links to the author's home-pages in WebBiblio. When the results are displayed, WebBiblio also provides links to the journals, publishers and editors' home-pages, if available, in the same manner. Achilles's advanced search engine site provides two different types in displaying the search results. It gives the option between 'headline menu' and 'full references' displays [Achilles 1996c], where 'headline menu' option lists the name(s) of the author(s) and the title, and the 'full references' option provides all the information about the reference including the keywords and abstract, if available. Different from that, WebBiblio lists the entire bibliographic information of the matching references in displaying the search results and provides the abstract and/or keywords of the references upon request. We believe that WebBiblio has a higher usability by providing more detailed information about the reference without including the abstract-keywords information, which fills the query results page easily.

To conclude, WebBiblio provides a better query composition interface with a richer set of capabilities compared to the Achilles' "The Collection of Computer Science Bibliographies" site. When the approaches used in displaying the search results are considered, both of the sites have some advantages over each other.

2.3 Jones' "Searching the HPB Bibliographies"

Another search engine site based on the Glimpse [Manber 1996] search technique is "Searching the HPB (Hypertext Bibliography Project) Bibliographies" web page created

by David M. Jones [Jones 1996]. In the query form of this site, the reference fields on which searching is supported are very limited. More specifically they are only author, title and abstract fields, and the queries based on these fields must share the same form field. Since a help utility is not provided, the set of logical operators allowed in query composition is not clear, although the usage of AND-operator is shown with an example next to the form field. A capability that the search engine of the site has, which WebBiblio does not support, is allowing misspellings in the search queries.

2.4 Kamath's "Quick-Search Database Systems Research Bibliography"

Mohan Kamath's bibliography search engine "Quick-Search Database Systems Research Bibliography" [Kamath 1996] is another bibliography publishing site on the WWW, which is conceptually closer to WebBiblio because its search engine is implemented by the creator of the site instead of being imported.

Kamath [1996] introduces the site as "This search is performed using an indexing/search engine designed and developed for bibliography files by Mohan U. Kamath using a combination of Perl and C programs. The total system consists of less than 800 lines of code and uses inverted lists [Rijsbergen 1996], multilevel indexing and other optimizations to perform competitively with other [Achilles 1996d] bibliography search sites equipped with text retrieval engines." [Kamath 1996].

On this site, the searching process is performed only on author, title, keyword, abstract, year, annotation and institution reference fields. The search query can only contain three keywords and they are combined with logical AND-operation. Other logical

operations are not provided. The query composition capabilities are slightly increased by allowing the user to restrict the searched keywords to be consecutive in the matching references [Kamath 1996].

When the search results are displayed, neither the abstracts and keywords of references nor a way to access this information is provided. In conclusion, Kamath's search engine provides a very limited set of query composition capabilities and query result display features compared to the ones provided by WebBiblio.

2.5 Summary

Besides these four sites, there exists many other sites on the WWW providing searching service over differing bibliographies which have relatively more limited set of query composition capabilities and simpler structures. At the same time, much research is being conducted and there are also many WWW sites under construction.

In conclusion, WebBiblio, with its user-friendly interface, easy-to-use query forms providing simple and advanced query composition capabilities, detailed and structured help utility, informative result display including abstracts and keywords upon request and other supporting pages is a unique site for publishing bibliographies globally on the WWW. With these characteristics, it has some advantages, as reviewed, over other sites on the WWW. Table 2.1, which also supports this conclusion, summarizes the comparison of characteristics of the bibliography publishing sites discussed in this chapter.

Table 2.1 Comparison of WebBiblio Features with Other Reviewed Systems

	First Search®	Achilles' (advanced)	Jones'	Kamath's	WebBiblio
General Availability	No	Yes	Yes	Yes	Yes
Number of Searched Fields	10	5 + 1[1]	3	7	17
Search in Abstracts & Keywords	Yes	Yes[2]	Only in Abstracts	Yes	Yes
Help Utility	Yes	Yes	No[3]	Yes	Yes
The Search Engine	Internal	External	External	Internal	Internal
Simple Query Syntax	No[4]	Yes	Yes	Yes	Yes
Support of AND, OR, NOT & Parentheses	Yes	Yes	Only AND	Only AND	Yes
Logical Operators among Form Fields	Limited	No	No	No	Yes
Range Operators in Numerical Fields	Limited	Limited	No Num. Field	No Num. Field	Yes
Support of Search Flags	Yes	Yes	Yes	Yes	Yes
Misspellings Allowed	No	No	Yes	No	No
Match Score	No	Yes	No	No	No
Display Abstracts & Keywords if Available	Yes	Yes	Yes	No	Yes
Links to Author, Editor, Publisher, Journal Pages	No	No	Links to Journals	No	Yes
Contribution Form Support	No	No	No	No	Yes

[1] The sixth field is called "Anywhere in the reference" and covers all the fields [Achilles 1996c].
[2] Not as specific query form fields; covered by "Anywhere in the reference" field [Achilles 1996c].
[3] Except limited form attached help.
[4] Requires field names as reserved words in the query.

CHAPTER 3 : WebBiblio SOFTWARE ENGINEERING

This chapter provides a general overview of WebBiblio's software engineering. It introduces the WebBiblio's development phases based on waterfall software life-cycle model [Schach 1996], which is mainly followed in this research.

First the requirements are specified based on the objectives of the study. Then the hardware and software environments are identified. Finally design specification and general implementation structure of WebBiblio are presented.

3.1 Requirements Specification

WebBiblio is designed and implemented as a software system for publishing bibliographies with rich set of features in an efficient, easily and globally accessible way. The objectives of the study are stated under this goal, and the requirements are specified to fulfill each stated objective.

1. *WebBiblio should be a globally available tool on the WWW (Objective 1).*

    - WebBiblio must be a WWW software system.
    - WebBiblio must not have any access restrictions. It must be globally available.
    - WebBiblio must have a WWW page that have links to the individual bibliographies published by the software system.

2. *WebBiblio should facilitate the publication of all possible elements of a bibliographic reference structure (Objective 2).*

    - WebBiblio must be able to publish any kind of bibliography. To achieve this, all the forms of the system must contain all possible elements of a bibliographic reference structure and all these elements must be processed by the system properly.

3. *WebBiblio should be well human engineered (Objective 3).*

    - WebBiblio must be an easy-to-learn, easy-to-use tool. None of its pages must contain irrelevant or rarely used information.

    - WebBiblio must have simple syntax rules and minimal number of special characters. These characters must consistently used in different components of the software tool.

    - WebBiblio must have a consistent user interface.

    - WebBiblio must meet needs of users from a wide range of computer skills and experience.

    - WebBiblio must have a main menu that is available at all pages of the system in the same format. An option on the main menu must be the link to the web page that contains links to the published bibliographies.

    - WebBiblio must enable users to submit their comments and recommendations to the developers and/or administrators.

    - WebBiblio must provide links to the related sites.

4. *WebBiblio should provide on-line assistance (Objective 4).*

    - A detailed help utility about all the WebBiblio parts must be provided to answer all possible questions of the users from a wide variety of backgrounds and computer skills.

    - Help must be written in clear, simple and unambiguous manner.

    - All WebBiblio parts, including the help utility itself, must be explained in detail.

    - The utility must be simple to use and to navigate. Like all other WebBiblio components, the main menu must be available on all of the pages belonging to the help utility. Besides that, it must have its own main menu on all of the pages for the navigation purposes.

    - The contents of the help must be well structured and organized. The user must be able to find the information s/he needs easily. The structure of the utility must be displayed clearly for easy navigation.

- When the help utility is launched the initial page must give the most general information about the environment and the basic guidelines to use the help utility.

- When possible, all the section contents must be supported with a rich set of informative examples and their explanations. Each field of the forms must be explained in detail in separate sections. If applicable, a set of examples about the usage of the field with their explanations and the syntax rules of the field written in a specific format must be added to the contents of the section. All publication types that the references in the bibliography may belong to, must be introduced with examples from the bibliography database in the sections of help describing the query and contribution forms.

- The contents of the help about the contribution form must be enriched with screen-shots containing already filled form samples with different type of contributions as guiding examples.

- Since WebBiblio is a globally used tool, the help utility must provide the opportunity to its users to submit their comments and bug reports if any.

5. *WebBiblio should provide advanced search capabilities (Objective 5).*

    - A query form must be provided to allow the user to compose search queries.

    - The form must be easy to use. The syntax must be minimal. The user does not need to type more than the search keyword into the related field for simple queries.

    - Each element of bibliographic reference structure must exist as a field in the form. Hence, the user can compose a query for any element of a reference structure. Additionally, the query form must have abstract and keyword fields for the related queries.

    - The main menu must be available.

    - A reset button must be provided to clean all the fields of the form.

    - Any combination of the form fields must be used to compose queries. Empty (unused) fields must not have any restriction on the searching process.

    - The user must be able to view all the references in the bibliography. This should be done by submitting an empty query form.

- For each field, both simple and advanced search queries must be syntactically provided. The logical ‘and’, ‘or’ and ‘not’ operations and parenthesization must be allowed in each field, if applicable, for advanced query composition. The syntax of each field must be clearly defined and be consistent with the syntax of fields belonging to the same field class. The operators having the same logical meaning must be same in all fields.

- In the fields where the search is based on string search, the search’s flag values ‘case sensitivity’ and ‘whole word’ must be manageable by the user and be part of the query composition. Again, in these string search fields, the user must be allowed to enter search strings containing more than one word or special characters, or to convert the string search into an exact match search.

- In the numerical query fields, the number ranges must be also accepted as query tokens as well as the individual numbers. The user must be able to omit the upper or lower boundary value of the range so that s/he can also compose queries meaning ‘greater than’ or ‘less than’, respectively.

- The ‘and’ operation is the default logical operation among the form fields. Besides this default operation, a logical ‘or’ operation between the form fields must be provided to expand the user’s query composition capabilities.

- In parsing the search queries, when a syntax error is detected, the search engine running behind the query form must terminate properly and display an appropriate and informative message about the cause and the field of the error. The engine must be tolerant when parsing the user queries. Syntactically correct, but redundant information, like extra parentheses, overlapping queries, redundant space characters, must be handled properly and must not terminate the searching process.

- When displaying the search results of the query entered through the query form:

    - When searching process is complete, all the matching references must be listed on a new page in alphabetical order of the author last names. For the references belonging to the same author(s), the order must be determined with respect to the publication year in descending order. If the years are the same, then the order is determined with respect to the reference titles.

    - This new page must contain the main menu of WebBiblio.

- The references must be displayed with the URLs of their authors, publishers and the publications in which they have been published, when these URLs are known.

- If available, the user must be able to view the abstract and/or keywords of any listed reference. Hence there must be three buttons attached for each reference: one is for the abstract, one for the keywords and one for both. When displaying the abstract and/or the keywords:

    - The requested information must be displayed on a new page with the information about the reference itself. This new page must have the main menu of WebBiblio.

6. *WebBiblio should be verified, validated and tested as much as possible (Objective 6).*

    - The functionality of all the components of the software system must be completely verified, validated and tested. First, module testing must be performed for each developed component on individual basis. Then, the entire system must be verified, validated and tested with the integration of the system components. Same approach must be also used recursively during the VV&T of the components that consist of sub-components and can be developed partially (for instance fields of query form, modules of the search engine, etc.).

    - Considering the worldwide range of WebBiblio users, incorrect usage of the system components must be expected. Hence, WebBiblio must be also tested with as many as possible incorrect inputs, and its appropriate responses for these inputs must be verified and validated.

    - A link must be provided for users to report any detected bugs.

7. *WebBiblio should provide abstracts and keywords of the sources cited by the bibliographic references (Objective 7).*

    - Abstracts and keywords must be considered as part of the references in the published bibliographies. A rich set of search capabilities must be also applicable with these fields consistent with the search capabilities over other parts of a bibliographic reference.

    - WebBiblio must enable the user to view the abstract and/or keywords of any bibliographic reference, if available. The user must be able to choose between viewing the abstract, the keywords or both.

8. *WebBiblio should respond to user queries as fast as possible (Objective 8).*

    - Response time efficiency must be the basic consideration when implementing the search engine. The time must be minimized in the searching process and information retrieval. The answers for the predictable queries must be kept in the files so that these queries can be answered without any search. The information must be retrieved from the data files directly without any search and for that purpose the relevant directory files must be used. Consequently, besides the response time efficient implementation of the engine, a set of special data files must be designed and created from the bibliography database to minimize the searching time.

    - Redundancies in the numerical queries, which may exist as a result of overlapping number ranges in the query, must be detected and cleaned before the searching process to prevent a time consuming redundant searching.

    - Upon user's request, the abstract and/or keywords information must be retrieved from the related files in a timely manner, more specifically directly without searching any data file.

9. *WebBiblio should be easy to administer (Objective 9).*

    - The routine administrative tasks must be simple and all of the steps of the procedures must be clearly defined.

    - These tasks must not require deep knowledge of system components.

10. *WebBiblio should be easy to maintain (Objective 10).*

    - WebBiblio must be easy to maintain and modify. All of the system programs must be very well documented. The variables must have meaningful names. The functions must be implemented in a loosely coupled manner and high cohesion must be achieved within the functions to increase the maintainability [Sommerville 1996b and Constantine 1979]. The high maintainability is an essential goal considering that the system administrator is not necessarily the system developer.

11. *WebBiblio should enable anyone to contribute to the published bibliographies (Objective 11).*

    - A contribution form must be provided to allow users to make contributions to the bibliographies.

- The form must be easy to use. The user does not need to write more than the information of the contributed reference itself. The rules that the contributor should follow must be simple, logical and clearly defined.

- All the possible elements of bibliographic reference structure must exist as a field in the form as well as the abstract and keywords fields. The contributor must be able to easily figure out which element of the reference to write in which field of the form.

- The form must ask also some information about the contributor. This is necessary for 'credits' page and for authentication purposes.

- When a contribution is made, all the required fields must be checked and if at least one of them is empty then the contribution must not be accepted and the contributor must be informed with an appropriate error message on a new web page. The contributed information must be parsed for the recoverable invalidities and when detected they must be fixed.

- After a successful contribution, the contributor must be notified and thanked with an appropriate message on a new web page.

- A reset button must be provided to clean all the fields of the form.

- The main menu of WebBiblio must be available on the form page, and also on the message pages.

- A ‘credits’ page must exist to acknowledge the developers and the contributors. This page must also serve as an encouragement for more contributions.

12. *A simulation verification, validation and testing bibliography database containing at least 500 references with abstracts and keywords should be developed, and WebBiblio should be used to publish it worldwide (Objective 12).*

    - A simulation verification, validation and testing bibliography database containing at least 500 references with abstracts and keywords must be developed. When developing this bibliography database:

        - A careful library search for related publications must be done. The most recent publications as well as the older ones from a wide range of sources (journals, proceedings, newsletters, theses, etc.) must be found and entered accurately into the database.

- The abstracts of the publications must be entered into the database together with the reference information by using an OCR (Optical Character Recognition) software.

- The keywords of the publication must be entered into the database. If they are not provided on the publication, they must be identified.

- The bibliography database must be developed in parallel with WebBiblio so that it can be used in the verification, validation and testing of the software product.

- The development of the database must be complete by the time when the development of WebBiblio is complete. WebBiblio must be used to publish at least this bibliography when it is publicized and become a globally available tool on the WWW.

- As part of DMSO research project, in addition to Simulation VV&T Bibliography, an Expert System/Knowledge-Based System VV&T Bibliography must be created. WebBiblio must publish this bibliography as well.

3.2 Hardware/Software Environment

This section introduces the hardware and software environments used for the development and execution of WebBiblio throughout the study. The hardware section gives information about the server of WebBiblio that is currently used. The software section introduces the operating systems, the software tools and DBMS used in the development as well as the programming languages used for the implementation.

3.2.1 Hardware

The WWW server of the WebBiblio is a DECpc running NEXTSTEP. It has a Pentium 90 processor, 32MB main memory and 4.3GB disk capacity.

3.2.3 Software

WebBiblio is designed, implemented and maintained in an environment running NEXTSTEP operating system. It is also the WWW server environment of WebBiblio. The bibliography databases are managed in Macintosh environment.

For editing the files containing the source codes, web pages and *Help* utility contents, the Editor application of the NEXTSTEP is used. The web pages of WebBiblio are created by using HTML 3.0 [Graham 1996]. WebBiblio is written in PERL [Quigley 1995] programming language and interpreted by PERL interpreter version 4.0 running under NEXTSTEP environment on Mach UNIX operating system.

Together with the development of software system, the bibliography databases currently published by WebBiblio are also created and improved. Hence, besides the development environments of WebBiblio, a DBMS is selected for the management of the bibliography databases, which is also introduced below and mentioned throughout the rest of the thesis. However, WebBiblio is not dependent on this DBMS. Any bibliography database can be published by WebBiblio independent of which DBMS is used as long as the information in the databases are provided in the format of the data files on which the search engine is operating.

The bibliography databases are managed by using FileMaker® Pro [Claris 1992] database management system. A HyperCard system using HyperTalk 2.0 [Winkler 1990] scripting language on Macintosh is developed to create the data files from the bibliography databases for the searching process.

OmniPage® Pro [Caere 1996] running on Macintosh is used as the Optical Character Recognition software for the import of publication abstracts from printed materials into the bibliography database.

3.3 WebBiblio Design Specification and Implementation

WebBiblio is a production software system that has been in use since July 1, 1996. It has been used to publish three bibliographies on the WWW. These bibliographies are: (1) the Simulation Verification, Validation and Testing Bibliography (created and developed based on Objective 12), (2) Expert System/Knowledge-Based System VV&T Bibliography, and (3) Software VV&T Bibliography. They can be accessed on the website http://manta.cs.vt.edu/biblio as shown in Figure 3.1.

Each bibliography published under this page is independent of each other. Hence, any number of bibliographies can be grouped in a web page and be published simultaneously by using WebBiblio by including the links on this web page. With this approach, bibliographic sites publishing related bibliographies can be created.

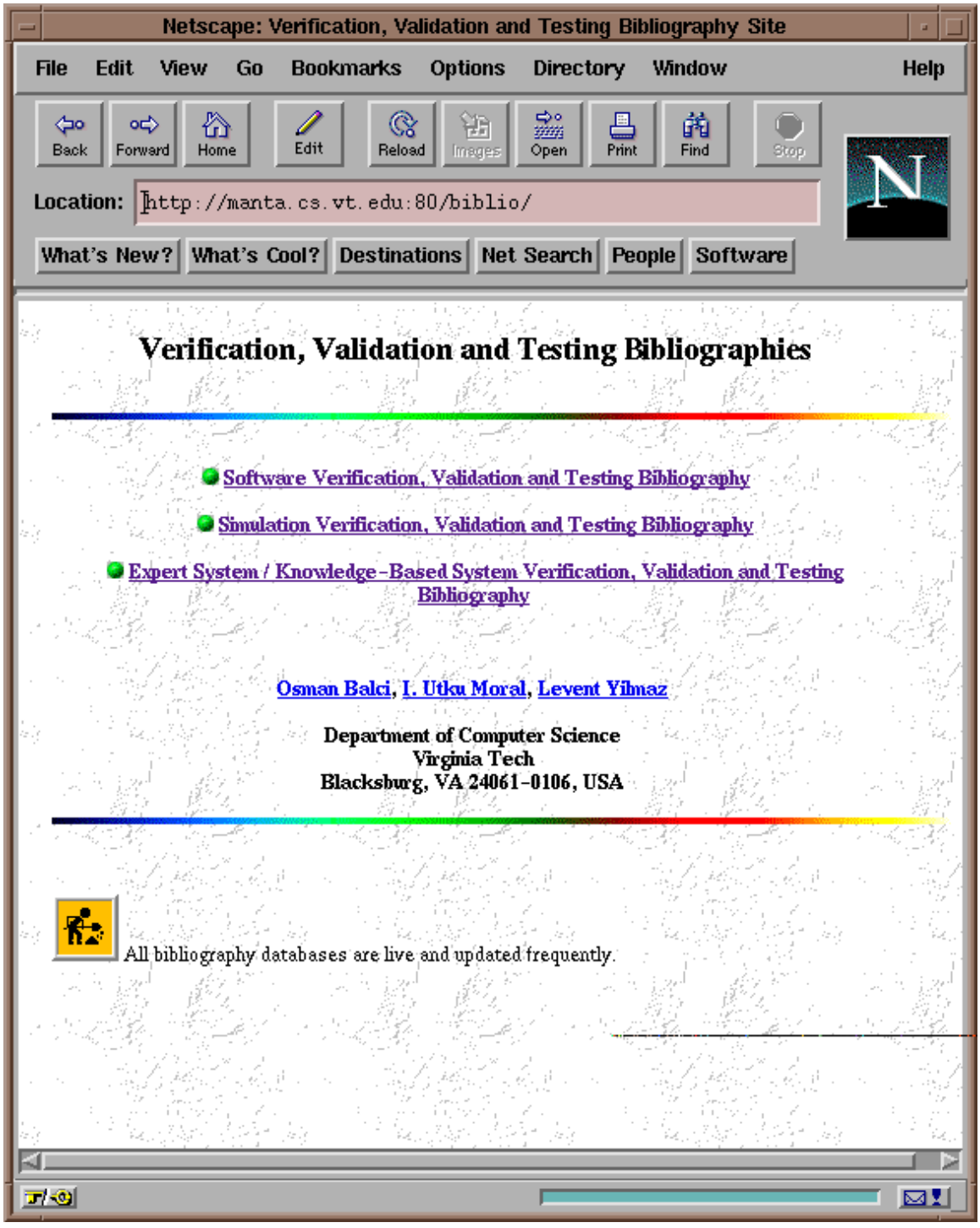

Figure 31 Web Page Containing Links to the Published Bibliographies

Each bibliography published by WebBiblio has the same structure, but different contents and database. The general implementation structure introduced in this section is common for all the bibliographies published. As depicted in Figure 3.2, each bibliography consists of *Query Form*, *Help*, *Credits*, *Contribution Form* and *Other Sites* pages. These pages are explained in detail in the following chapter.

The main menu of WebBiblio contains links to these five pages as well as the web page containing links to the other published bibliographies. This menu exists consistently on all WebBiblio pages providing an easy navigation between the system components.

When a bibliography is selected by the user from the web page presenting the published bibliographies, initially *Query Form* of the bibliography is downloaded. The searching process is activated by submitting a query using this form. The answer to the query is computed by the program *bibSearch.pl*, the search engine, by using the data files belonging to this bibliography, which are created from the bibliography's database.

After the query results are listed, the abstract or keywords requests are handled by the program *abskw.pl*, the abstract-keywords provider, which also uses the data files to retrieve the requested information. Similarly, the program *contribute.pl*, the contribution handler, is executed when a contribution is submitted using the page *Contribution Form*, and the contribution is sent to the special account *biblio* at WebBiblio's server domain *manta.cs.vt.edu* as an e-mail. All these other components of the system are also shown in Figure 3.2 and explained in detail in the following chapter.

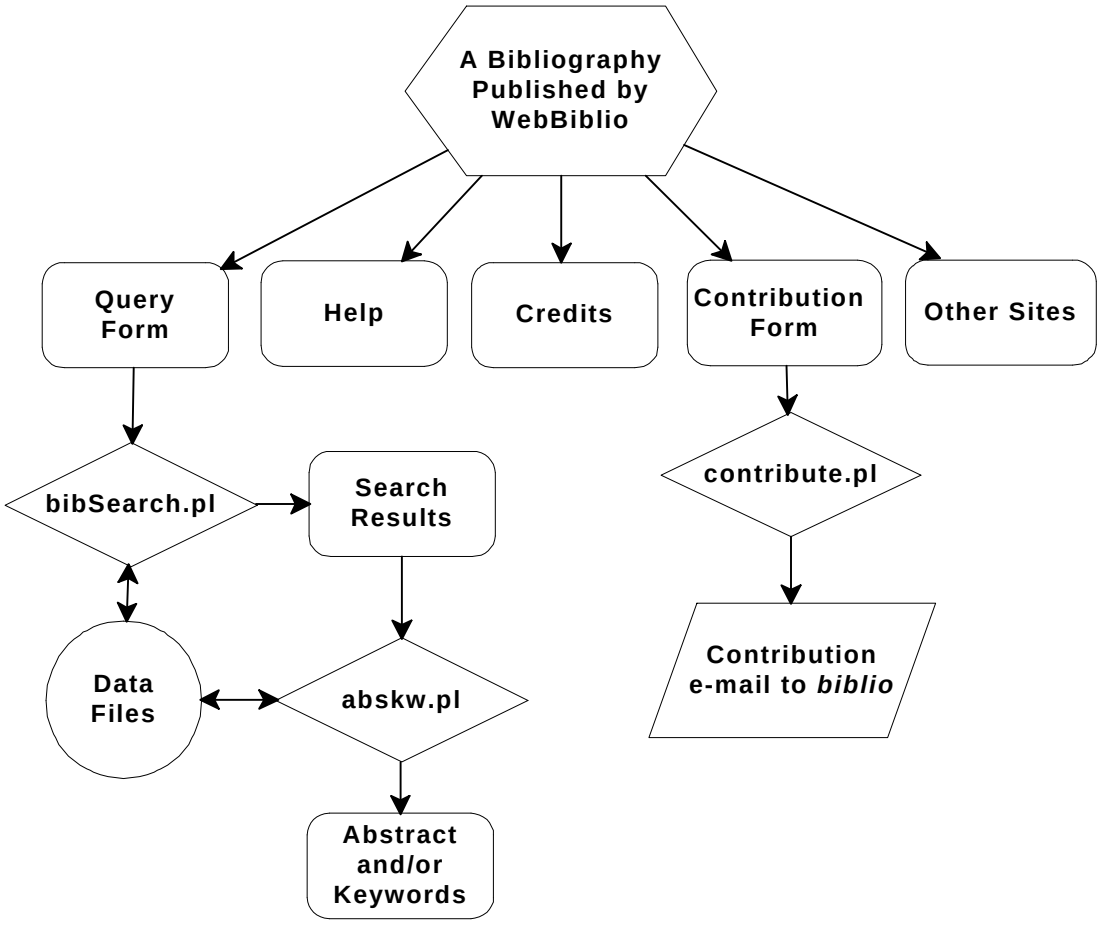

Figure 32 WebBiblio Architecture

CHAPTER 4 : WebBiblio USER'S GUIDE

This chapter describes the components and features of WebBiblio in detail. It focuses on each component of the software system one by one. For every component, the chapter explains its purpose and usage, introduces its structure and lists the features provided by the component. It classifies similar components to simplify the explanations. It also explains the usage, capabilities, and syntax of each form field of two forms in detail. The descriptions and explanations are supported with various examples and screen-shots.

Section 4.1 describes *Query Form*. It explains the logic among the form fields and introduces the classification of the form fields. Then all these field classes are described in detail and their syntax is presented. Finally, the section explains the search engine and abstract-keywords engine running behind *Query Form*. Section 4.2 introduces the classification of the data files, on which the searching process is performed, and describes the usage and structure of each data file class in detail. Section 4.3 contains the information about the bibliography databases and summarizes how they are created. Section 4.4 explains *Help* with its characteristics and structure. Sections 4.5 and 4.6 introduce *Credits* and *Other Sites* pages respectively. Finally, Section 4.7 describes the *Contribution Form* and its use, and explains the contribution engine running behind the form.

4.1 Query Form

Query Form is used to compose queries to perform searches over the bibliography databases. The form contains all reference form fields for the most advanced search

queries. Based on Objective 2, all parts of a bibliographic reference information belonging to any of the publication types can be found as fields of the form. Figure 4.1 shows *Query Form* launched under the Simulation VV&T bibliography.

The general logic of query composition is based on a simple approach. Every field of bibliographic reference information as well as keywords and abstract are provided as a field of *Query Form*. Query composition is performed by entering qualifiers in these fields. Each qualifier entered in a field is processed as a search query over the corresponding reference information field. The more qualifiers entered into the fields, the more specific is the search query, since the references in the result have to match to every query on the form. For instance, entering 1971 in *Year* field displays the list of all references with publication date 1971. To make this query more specific, all the user needs to do is to improve the query by providing more qualifiers. For instance, the user may restrict the search results also based on authors. For example, in addition to the year of 1971, entering Smith in *Author Name* field results in the list of all references with publication date of 1971, and a (co)-author last name of Smith".

In summary, every qualifier entered in a form field serves as a search query over the corresponding reference information field. The user does not use a special syntax and reserve words to address the reference field to perform the search operation. Entering qualifiers into the corresponding fields is sufficient. This approach for query composition is easy to learn and to use. The syntax is minimal - for simple queries almost none - and its logic is simple. Hence, this approach is convenient for a globally accessed tool serving users having a wide variety of backgrounds and computer skills. Based on this

consideration, achieving a high level of simplicity guides the design of *Query Form* and determination of each field's syntax.

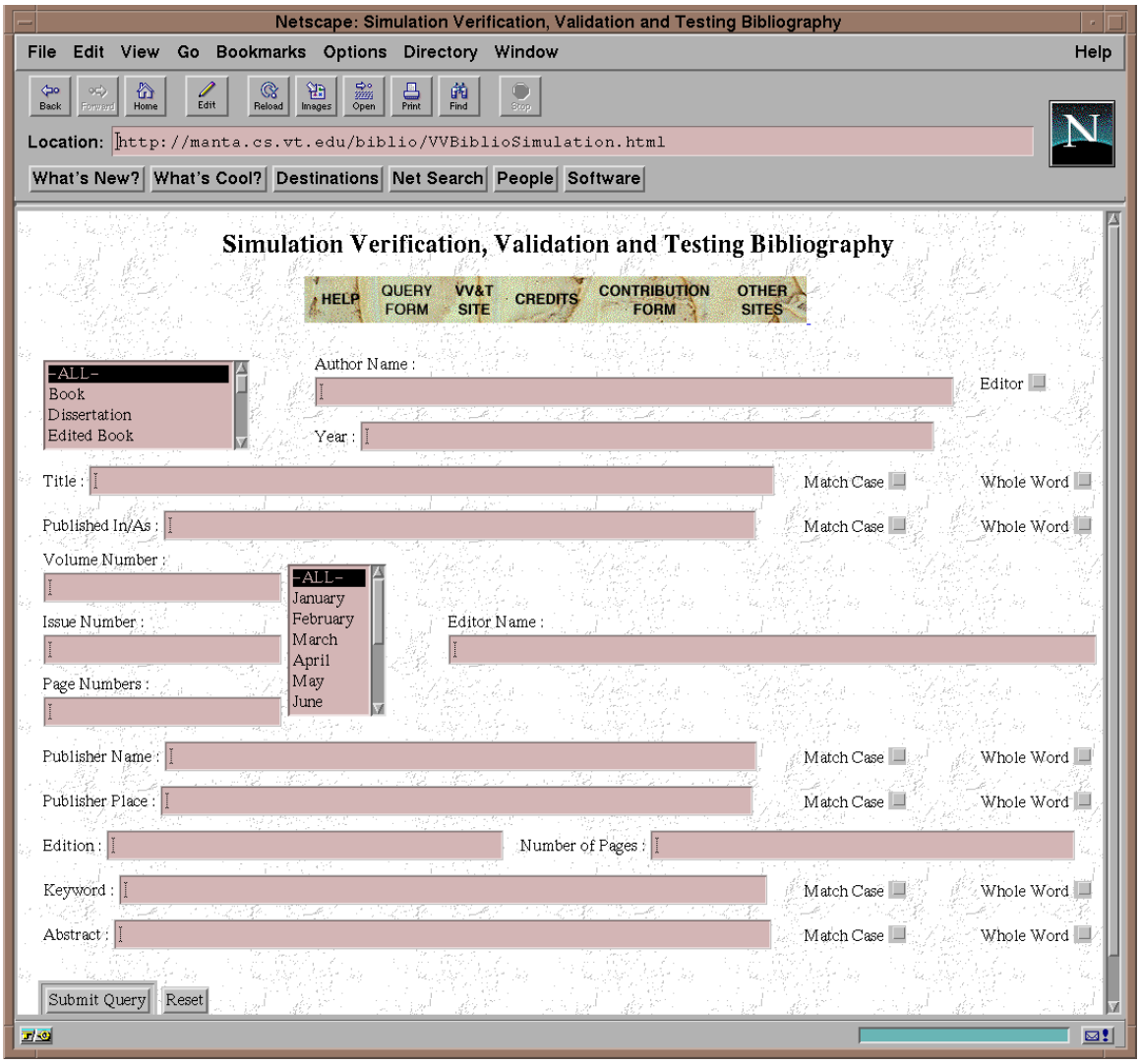

Figure 4.1 WebBiblio*Query Form*

By default, the fields of the query form are combined with logical AND-operation (in the second example, the matching references are published in 1971 'and' Smith is the last name of at least one of the authors.) Besides providing a query syntax within a form

field for more advanced search queries, to increase the query capabilities of the form even further, the logical OR-operation among the form fields is also provided. The detailed information about the logic among the form fields is given in Section 4.1.1.

After composing a query, the submission is performed by clicking on *Submit Query* button. Like many forms on the WWW, *Reset* button is provided to clear the entire form when clicked.

4.1.1 Logic among the Form Fields

AND is the default logical operation among the form fields. Queries entered in the fields of the form can be considered as logical expressions, which return a logical value. The entire search result is determined by combining the logical values received form each field of the form by applying logical AND-operation. If a field is empty, or passes the -*ALL*- value if it is a list menu, then this field returns the logical value *true* (*one*). Hence, this field has no effect on the logical operation among the fields, and consequently no restriction on search results. If the form is submitted with all its fields empty, then the logical expression of the entire form returns *true*, which means empty query, or no restriction. That causes the listing of all the references in the bibliography database.

The processing of a non-empty field, if the query is syntactically correct, returns the set of references matching to the query, which can be also an empty set when there is no matching reference. The empty set is equivalent to the logical value *false* (*zero*). Hence, unless the logical OR-operator between the fields is used, a form field returning *false* makes the entire form return *false* since a logical AND-operation between *false* and

any logical expression always returns *false* (i.e. if there is no reference in the bibliography matching to the query of a form field, then there is also no reference matching to the entire query.)

After the queries in non-empty fields are processed, the result of entire query is obtained by computing the intersection of the sets that are returned by these fields as search results. By computing the intersection of the sets, the set operation that is equivalent to the logical AND-operation is applied.

In the expense of increasing the response time, to expand the query composition capabilities provided by *Query Form*, logical OR-operation among the form fields is provided. This feature increases the response time of WebBiblio in cases when there is no matching reference to the user's search query. An empty set returned as the result of processing a field's search query does not mean the termination of the entire search, which would be the case if only AND-operation were allowed among the fields (since intersection of any set with empty set is also empty set, the result of entire search would be empty set, which means no matching reference.) Since the possibility of an OR-operation used among the remaining form fields exists, the search engine cannot terminate immediately by reporting that there is no matching reference to the query.

The conversion of the logical operation between the form fields from default AND-operation into the OR-operation can be made by entering the pipe character (|) as the first non-blank character in the field. This character makes the search result set of that field be united, not intersected, with the search result set of the previous fields. Note that the union operation is the equivalent of logical OR-operation among the set operations.

The comparison of two sample queries below explains the difference in interpreting these two operations. The first sample query is composed by entering Jones in *Author Name* field and 1996 in *Year* field. This query displays the list of all references with publication year of 1996, *and* a (co-)author last name of Jones. If this query is modified by adding the pipe character into *Year* field as the first non-blank character (the query in *Year* field becomes | 1996) then the new query displays the list of all references with publication year of 1996, *or* a (co-)author last name of Jones. For instance, although a book written by A. Jones in 1974 or a thesis written by U. Moral in 1996 match to the second query, they don't match to the first query.

Figure 4.2 displays a more advanced search query that uses four different fields of the form. The results of the query after running it over Simulation VV&T Bibliography is shown in Figure 4.3.

If a form field does not contain any other character besides the pipe character then the pipe character is ignored, and the field is considered as it is empty. This situation is assumed as harmless user error. It is simply neglected so that the searching process can continue instead of terminating with an error message.

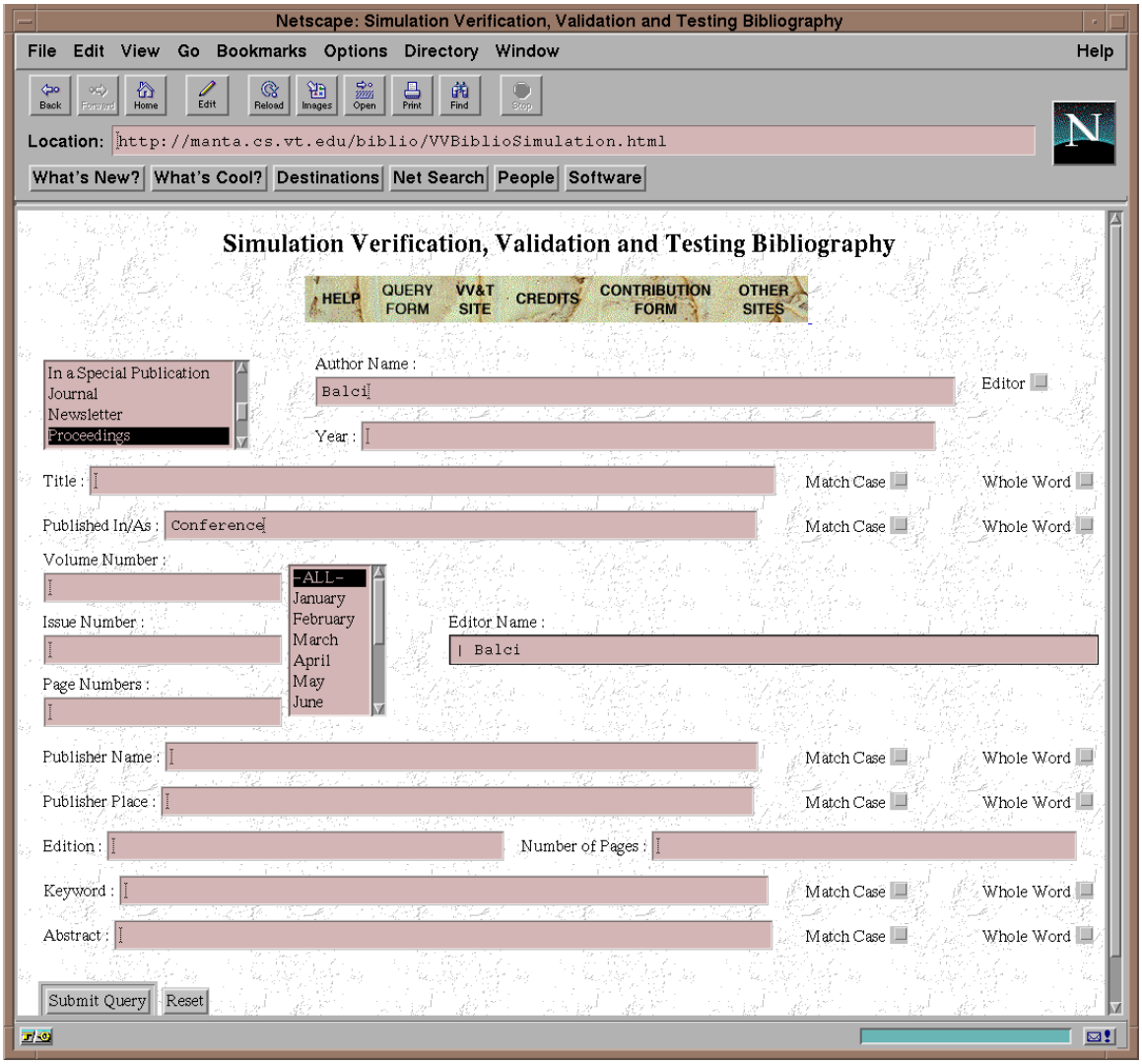

Figure 42 Search Query Example

Although capability of changing the default operator into OR-operator between two form fields does not provide a complete freedom to compose any logical structure between the form fields, it still increases the search capabilities of the user considerably. For instance, since the empty fields have no effect on the search, the search results of two fields can be united by leaving all other fields blank and typing the pipe character as the first non-blank character in the successor one. As another restriction in using OR-

operation among the form fields, the checkbox field, *Editor*, and the two list menu fields, *Publication Type* and *Month*, cannot be combined with logical OR-operation, since their structure does not allow typing of any characters.

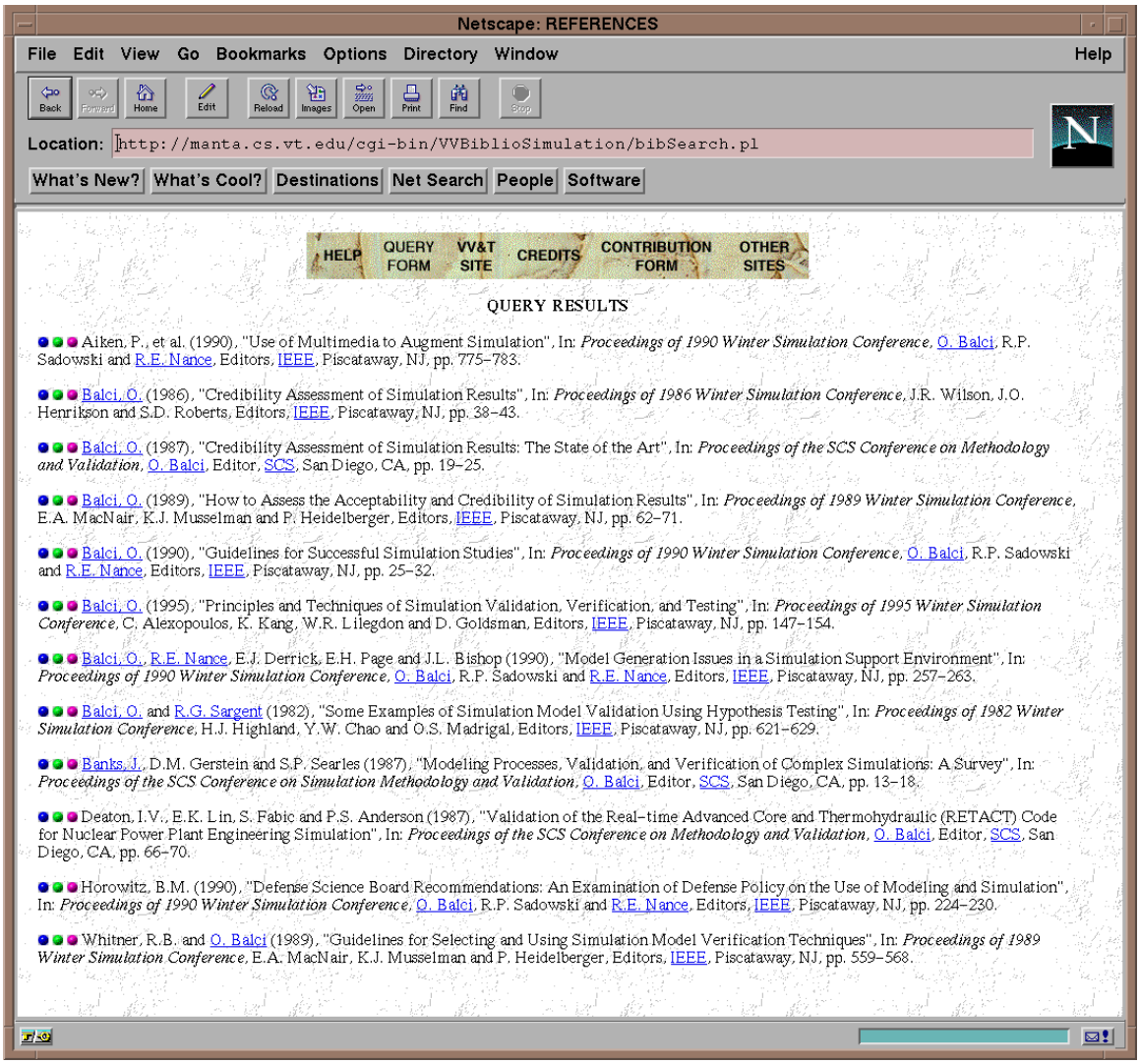

Figure 43 Results of the Query in Figure 4.2

When the logical operations among the form fields are applied, the form fields are processed in the order of: *Publication Type*, *Author Name*, *Editor*, *Year*, *Title*, *Published*

In/As, Volume Number, Issue Number, Month, Page Numbers, Editor Name, Publisher Name, Publisher Place, Edition, Number of Pages, Keyword and *Abstract* .

4.1.2 Logic and Syntax of the Form Fields

According to their logic and syntax, the fields of the form are subdivided into six classes that are described below in detail.

4.1.2.1 List Menu Fields

The fields *Publication Type* and *Month* belong to this class. Since the set of the values that the references in the database can have in their publication type and month fields are well defined, the usage of the list menu structure for these fields is possible. This gives an opportunity to provide the user a simpler, more user friendly, less error-prone query composition interface. By displaying all the available options as a list, many possible user errors, mainly syntax errors, are prevented.

All the types of the publications and the months of the year are known, and they are listed in the list menus of the fields. Query composition on these fields are performed by selecting any combination of the listed items. For instance, if the user selects the item Book on the *Publication Type* list menu, then the query displays the list of all references of type book.

When multiple selections are made then these selections are combined with logical OR-operation. For example, if the user selects the item Book and also Thesis on the

Publication Type list menu, then the new query displays the list of all references of type book *or* thesis.

The logical AND-operation is not provided among the multiple selections, since it does not have any practical usage. Every reference in the bibliography database has exactly one publication type, consequently AND-operation between two different publication types always results in an empty set. Except some references (less than 1% of the references), every reference in the database has at most one publication month, consequently the AND-operation for this field is useless as well.

The default selection for list menu fields is -*ALL*-. This is equivalent to leaving the field empty, which means no query, or no restriction. Depending on the capabilities of the browser, the item -*ALL*- in the lists are highlighted to indicate that it is the default selection.

A semantic mistake that the user can make while using list menu fields is to make a multiple selection from the menu including also the item -*ALL*-. This situation can be described as the computation of the union set of some sets including the universal set, hence the result is always universal set, or in other words logical *true*. This is considered as harmless user error, and it does not cause the termination of the searching process with an error message. The other selections are ignored and only the item -*ALL*- is taken into consideration. Consequently, the field is assumed as empty, and no restriction is applied on the searching process based on this field.

Selecting all the publication types in *Publication Type* field is equivalent to selecting -*ALL*-, since every reference in the database has one and only one publication

type. This is not the case for *Month* field, since some references do not have a month entry. Consequently, the query of selecting all the months on the list returns the list of all references with non-empty month entry, which is not logical *true* that selecting of *-ALL-* would return.

4.1.2.2 Name Query Fields

The fields in this class are used to compose queries based on names. The fields *Author Name* and *Editor Name* belong to this class. To provide a higher flexibility, the user is allowed to compose name queries in three different formats: initials, last name and full name.

In the “initials” format, the user only enters the initials of the author or editor excluding the last name as the search query. The initials are entered in capital letters and separated by periods. For instance, if A.A. is entered in *Author Name* field then the query produces the list of all references with (co-)author initials of exactly A.A.. Note that, for example, a publication written by A.A.B. Smith does not match to the query since the initials must be exactly A.A..

To compose queries in the “last name” format, the user enters only the author’s or editor’s last name as the first letter in upper case the rest in the lower case. Similar to the first format, for proper search query the last name of the author or editor must be entered completely, since the search engine looks for an exact match for the name queries in all three query formats. For instance, if Jones is entered in *Author Name* field then the query returns the list of all references with a (co-)author last name of Jones.

The “full name” format is the combination of the first two one and can be used to compose more specific search queries. Continuing with the examples, if A.A. Jones is entered in *Author Name* field then the query displays the list of all references with (co-) author name of A.A. Jones.

To increase the capabilities of *Query Form* and allow more advanced search queries, logical operations among the name queries are provided. These are binary AND and OR-operations and unary negation operation. The binary operations have equal precedence and they are evaluated from left to right. The unary operation has higher precedence than the binary operations. To change the precedence order, paranthesization can be used. Table 4.1 displays the query syntax of Name Query Fields specifically in BNF (Backus-Naur Form), where the double quotation marks are used for the terminals that are explained instead of being listed (same approach is used for all syntax tables throughout the thesis).

Table 4.1 Query Composition Syntax of Name Query Fields

<NameQuery>	::=	<NameExpression>|| <FormOR_operator><NameExpression>
<NameExpression>	::=	<Name> || (<NameExpression>) || <NameExpression><OR_operator> <NameExpression> || <NameExpression><AND_operator> <NameExpression>
<Name>	::=	<NameFormat> || <NOT_operator><NameFormat>
<NameFormat>	::=	<FullName> || <LastName> || <Initials>
<FullName>	::=	<Initials><LastName>

<Initials> ::= <Initial>. || <Initial>.<Initials>

<Initial> ::= "The initial of the author's name, capital letter"
<LastName> ::= "Author's last name,first letter is upper case, others lower case"

<FormOR_operator> ::= |
<AND_operator> ::= &
<OR_operator> ::= |
<NOT_operator> ::= !

The meaning of the AND-operation among the name queries is that both of the names must be in the author (or editor) set of the matching references. For instance if A.A. & Jones is entered in *Author Name* field as the query, then it displays the list of all references with at least one (co-)author having initials A.A. *and* also at least one (co-)author having the last name Jones. Note that the query A.A. & Jones is not equivalent to the query A.A. Jones, which is given as an example above. For instance, although a publication written by A.A. Jones matches to both queries, a publication written by A.A. Smith and P. Jones matches only to the first query.

Similarly when an OR-operation is used among the two name queries, at least one of these names must be in the author (or editor) set of the matching references. For instance if Smith | Jones is entered in *Author Name* field as the query, then the query returns the list of all references with a (co-)author last name of Smith *or* Jones.

Unary negation can be used to exclude a certain author or editor name in the author or editor set of the matching references. For example the query ! Jones in *Author Name* field returns the list of all references *without* a (co-)author last name of Jones.

Except the interpretation of negation operation, every concept is identical for the *Author Name* and *Editor Name* fields. The interpretation difference of negation operation between these two fields is based on the difference of their universal sets. Since every reference in the bibliography database has at least one author, when the negation operation is applied, the universal set is the complete set of the references. This is not true for *Editor Name* field because of the references in the database without an editor name entry. That makes the universal set of the negation operation for *Editor Name* field a subset of all the references, in other words the set of all references with editor name entry. For instance, if the query above, ! Jones, is entered in *Editor Name* field instead of *Author Name* field, then it returns the list of all references with an editor name entry and without a (co-)editor last name of Jones.

4.1.2.3 Editor Checkbox

Editor field is the only checkbox of *Query Form* that is a field by itself. As being a checkbox, the field has two possible values, checked or unchecked. Unchecked is the default value and has no impact on the searching process. When the box is checked, then the resulting references are restricted to the ones that refer to the publications whose authors are also their editors. The value of the checkbox is switched by clicking on it. As a result of its structure no syntax rules are applicable for this field.

4.1.2.4 Numerical Query Fields

The form fields used to compose numerical search queries belong to this class. These are *Year, Volume Number, Issue Number, Edition* and *Number of Pages* fields.

In *Year* field, to simplify the query composition, year queries in two digits are allowed. Consequently, year queries can be entered as two digits or four digits. If the first two digits are omitted, the year is considered in the 20th century. When using the bibliography in the future, to refer to the references of the new century all four digits should be used.

Besides the individual numbers, number ranges in the query are also allowed to increase the querying capabilities of the user. The ranges can be entered in three different formats: (1) the lower and the upper boundary values of the range, (2) only the lower value, and (3) only the upper value. In all the range formats, the range operator dash (-) is used to distinguish the range queries from the individual number queries.

When using the first format, the range operator is placed between two boundary values. For instance, the query 1971-74 entered in *Year* field uses this format and it displays the list of all references with a publication year *between* 1971 and 1974 *including* 1971 and 1974.

In the second format the operator follows the boundary value, and in the third format it leads to the boundary value. If we change the query above into 1971- then it uses the second format and it displays the list of all references with a publication year of 1971 or later. Similarly, the query -74 uses the third format and it displays the list of all references with a publication year of 1974 or earlier.

Individually, or as a range, more than one numerical query can be entered in these fields to compose more advanced search queries. Multiple numerical queries are separated with comma. The comma between these queries can be considered as the logical OR-

operator, since the result of entire query is obtained by computing the union set of each individual query's result set. The meaning of multiple queries can be explained by the example query 4, 6-10 entered in *Volume Number* field, which displays the list of all references with the volume number of 4 *or* any number between 6 and 10 including 6 and 10.

All references in the database have only one or zero numerical entry in their numerical fields. As a result of this, a numerical query containing a logical AND-operation always results in empty set. Hence, AND-operation among the numerical queries is useless, and it is not provided for the form fields belonging to this class. The logical negation is also not provided, since it does not introduce a new functionality. By using the range and multiple query capabilities, the equivalent query can be composed without the need of the logical negation operator. For instance, an imaginary query 1975-1985 and not 1982 in *Year* field has the equivalent 1975-81, 83-85, which is also syntactically correct. Table 4.2 shows the query syntax of the Numerical Query Fields in BNF format.

The wide flexibility provided to the user in numerical query composition also brings a problem with: redundancy. For example, the query 1970-, 90-96 in *Year* field is syntactically correct, although the second sub-query is redundant. Redundancy in queries causes unnecessary repetition of the costly searching process. Hence, detection and cleaning of the redundancies in queries is an essential part of the searching process for achieving a low response time (Objective 8).

Based on this consideration, the search engine performs redundancy detection and cleaning on Numerical Query Fields queries before the searching process. First, it converts

all the queries into number intervals. The individual numbers are also handled as the number intervals with the same start and end points. Following that, the number intervals are sorted and when an overlap is detected these two (or three in the case of the new interval fills the gap between two intervals) intervals are combined into one new interval. With this process, the redundancy is cleaned with a relatively low cost compared to the costly and redundant searching process that would be performed by the search engine. Besides that, during the process the intervals are sorted, which is an essential step to increase the execution efficiency of the searching process. The insertion sort algorithm is used for the sorting of the number intervals. When the number of the intervals is considered (in most cases, less than 5), this simple algorithm with little overhead computation and without a need for a special data structure appears as the best sorting algorithm for the problem.

Table 42 Query Composition Syntax for Numerical Query Fields

<NumberQuery>	::=	<NumberExpression> || <FormOR_operator><NumberExpression>
<NumberExpression>	::=	<NumberEntry> || <NumberEntry><Separator> <NumberExpression>
<NumberEntry>	::=	<Number> || <NumberRange>
<NumberRange>	::=	<Number><RangeOperator><Number> || <RangeOperator><Number> || <Number><NumberOperator>
<Number>	::=	<Year> || “Number in digits”
<Year>	::=	"4-digit year" || "2-digit year"

<FormOR_operator> ::= |
<Separator> ::= ,
<RangeOperator> ::= -

4.1.2.5 String Search Query Fields

The fields in this class are used to compose the queries based on the string search over the related information. In contrast to the fields in Name Query Fields, where the searched strings must exactly match with the related information of the matching reference, in these fields the search is based on the string search. The matching references have to contain the searched strings in their corresponding fields, but they do not need to be exactly same. The fields *Title*, *Published In/As*, *Publisher Name*, *Publisher Place*, *Keywords* and *Abstract* belong to this field class.

Each field belonging to this class has two checkboxes next to itself on *Query Form*. These checkboxes determine the characteristics of the string search that is going to be performed for the query composed in the corresponding fields.

The checkbox *Match Case* controls the case sensitivity of the search. When it is not checked, which is also the default value, the search is case insensitive. When it is checked, then the case of the search string and the case of matching strings have also to match.

The checkbox *Whole Word* determines the value of the other search flag. When it is not checked, which is also the default value, the search string is checked whether it is contained in the searched information or not, not necessarily as a complete word. In other words, the matching references have the search string in their related field information as

an entire word or as a part of a word. When the checkbox is checked then the search is restricted to only entire words, hence all matching references contain the search string as a whole word in their searched fields.

Consistent with all the rest of the form, the default values of the checkboxes are the most general values. By checking as many of them as wanted, the user can make his/her search query more specific. The values of the checkboxes are ignored, if their corresponding fields do not contain any query.

Same as Name Query Fields, to expand the functionality of *Query Form* and to provide the user a wider set of capabilities, logical AND and OR-operations, negation operation and parentheses are supported also in String Search Query Fields. Table 4.3 on page 49 shows the query syntax of these fields in BNF format.

When the queries contain more than one search string combined with logical operations, then the flag values of the checkboxes are applied to the searching process of all the strings in the query.

The meanings of the logical operations used in these fields are also similar to the meanings of logical operators used in Name Query Fields. For example, the query life & cycle in *Title* field returns the list of all references with titles containing the strings life *and* cycle. Addition to this query, for instance, if *Whole Word* box is also checked then the new query returns the list of all references with titles containing the *words* life *and* cycle. If the query is modified further by replacing AND-operation with the OR-operation then it returns the list of all references with titles containing the words life *or* cycle.

The meaning of negation operation is slightly different for *Title* field compared with the other fields belonging to this class. This is caused by different universal sets used for negation operation. Every reference in the database has a non-empty title entry, which is not true for other fields on which the string search is applied. For instance, the query ! object in *Title* field returns the list of all references with titles *not* containing the string cycle, although the same query in *Keywords* field returns the list of all references with *keywords entry*and none of these keywords contain the string cycle.

In addition to the query syntax of the Name Query Fields, String Search Query Fields allows the usage of double quotation marks in the query composition to increase the search capabilities provided to the user further. According to the interpretation of the double quotation marks in the query, the fields of this class are divided into two subclasses.

For the *Title*, *Published In/As* and *Abstract* fields, double quotation marks can be used to enter search strings with more than one word, in other words, search strings containing space character. Usage of the double quotation marks is necessary for this case, since the space characters are ignored during the parsing of the query. The user can also enter the special characters, like & and |, between the quotation marks as the characters of the search string, which would be considered as logical operators of the query otherwise. For instance, the query “life cycle” in *Abstract* field returns the list of all references with abstracts containing *exactly* the string life cycle. Note that the query life & cycle is not equivalent to the query “life cycle”. The former one is less restrictive. For instance, the phrase ‘cycle of life’ in an abstract matches to the first one, but not to the second. For

these three fields, the values of the checkboxes are considered also for the search strings entered in double quotation marks.

For *Publisher Name, Publisher Place* and *Keywords* fields, the double quotation marks are used to convert the string search into the exact match search. When a string is entered in double quotation marks as the query, the matching references have exactly the same string as the data in their related fields. For instance, if the query credibility in *Keywords* field returns the list of all references whose one of the keywords *contain* the string credibility. If the same query is converted into "credibility" then it returns the list of all references whose one of the keywords *is exactly* credibility.

In these fields, since the usage of double quotation marks converts the searching process into an exact match search including the case of characters, the values of the checkboxes become obsolete. Hence, they are not considered for the search strings in the query that are entered in double quotation marks.

Table 43 Query Composition Syntax for String Search Query Fields

<StringSearchQuery>	::=	<StringSearchExpression> || <FormOR_operator> <StringSearchExpression>
<StringSearchExpression>	::=	<StringSearchString> || (<StringSearchExpression>) || <StringSearchExpression><OR_operator> <StringSearchExpression> || <StringSearchExpression><AND_operator> <StringSearchExpression>
<StringSearchString>	::=	<StringType> || <NOT_operator><StringType>
<StringType>	::=	<String> ||

<DoubleQuotation><DoubleQuotationString>
<DoubleQuotation>

<String> ::= "String of letters and digits"
<DoubleQuotationString> ::= "String of characters "

<FormOR_operator> ::= |
<AND_operator> ::= &
<OR_operator> ::= |
<NOT_operator> ::= !
<DoubleQuotation> ::= "

4.1.2.6 Page Number Field

Page Number field can be used to compose queries to search the database over the page number field of bibliographic reference information. To increase user's search capabilities, the query syntax of the field allows three different formats for query composition: beginning page number, ending page number, and both.

When the "beginning page number" format is used, then the search is based on the beginning page number of the cited source in its publication. To use this type, the range operator - (dash) is entered following the page number. For instance the query 41- returns the list of all references with *beginning* page number of 41. If the operator is used before the number then it stands for the "ending page number" format, and the search is based on the ending page number of the cited source in its publication. Hence the query -41 returns the list of all references with ending page number of 41. In the third format, both page numbers are entered. For example the query 25-41 has this format and it returns the list of all references with beginning page number of 25 *and* ending page number of 41.

Multiple page number queries are allowed in the field. Same to Numerical Query Fields, these multiple queries are separated with comma, which is considered as the logical OR-operator. For instance the query 25-41, -44 returns the list of all references with beginning page number of 25 and ending page number of 41, *or* ending page number of 44 whatever the beginning page number is. Considering the fact that every reference in the database has only one or zero page number entry, the logical AND-operation between the page number queries is not provided, since it would be useless by always giving an empty set as the operation result. Table 4.4 shows the query syntax of the Page Number Field in BNF format.

Table 4*4* Query Composition Syntax for Page Number Field

```
<PageNumberQuery>        ::=  <PageNumberExpression> ||
                              <FormOR_operator>
                              <PageNumberExpression>

<PageNumberExpression>   ::=  <PageNumberEntry> ||
                              <PageNumberEntry><Separator>
                              <PageNumberExpression>

<PageNumberEntry>        ::=  <StartingPageNumber><RangeOperator> ||
                              <RangeOperator><EndingPageNumber> ||
                              <StartingPageNumber><RangeOperator>
                              <EndingPageNumber>

<StartingPageNumber>     ::=  "number in digits"
<EndingPageNumber>       ::=  "number in digits"

<FormOR_operator>        ::=  |
<Separator>              ::=  ,
<RangeOperator>          ::=  -
```

4.1.3 The Search Engine: bibSearch.pl

The program *bibSearch.pl* is the main component of WebBiblio. It is basically the parser and the search engine running behind *Query Form*. It receives the user's search queries as input, parses them, and performs the searching process for syntactically correct queries by using the data files. It displays the search results, or an appropriate message if none of the references matches the query or the query contains a syntax error.

As first step, the program defines its constants and executes the required code for the creation of a new web page to display the search results. Second, it places the main menu of WebBiblio to the top of the new page, since the page must have this menu at the top like all other web pages of WebBiblio. Third, it processes the input string and determines which search query belongs to which field of the form. As the final step before parsing of search queries, the program declares and initializes the global variables.

The program parses the fields sequentially in the order they appear on *Query Form*. This provides high maintainability for the program. Initially, the search result is logical *true*, which means all the references in the bibliography. Then every form field is processed one by one. Empty fields are skipped without changing the search result. Hence, empty query form returns all the references in the bibliography database. The queries in non-empty fields are parsed, the search operation is performed. When the search result of an individual field is computed, the search result of the form is updated by combining the new result. After processing the last field, the result becomes final and the matching references are displayed. If a syntax error is detected, the program immediately terminates with an informative error message. Figure 4.4 contains an example query result display created by *bibSearch.pl* based on query 1979 entered in *Year* field and searched over the

Simulation VV&T Bibliography. The buttons next to each reference can be used to view the abstract and/or keywords of the reference. The usage of the buttons is explained in Section 4.1.4 on page56 in detail.

The details of search operations performed for each field and each type of data files, and combination of field search results are explained previously in this section and in Section 4.2 on page58.

Figure 44 Display of the query results by *bibSearch.pl*

To fulfill the high maintainability objective (Objective 10), *bibSearch.pl* contains a module for the manipulation of each form field. The code is repeated with necessary minor changes for the fields belonging to the same field class. Hence the main body of the code can be considered as the collection of field specific modules. The segments are written sequentially according to the order of their appearance on *Query Form*. Consequently, the modules performing the searching process for form fields are not nested; the modules are independent of each other. This provides high flexibility in modifying the program. Any change in *Query Form*, like adding or deleting a field, expanding the functionality or changing the syntax of a field can also be easily implemented in the program.

Similarly, the modules of *bibSearch.pl* possess minimum coupling to increase the program's maintainability [Sommerville 1996b]. Except the low level operation modules (for instance, the module that computes the intersection of two input sets), each program module is called by the minimal number of the field specific independent program modules. For instance, if manipulation of a form field needs a module that is similar to an existing module except a few minor changes, a new module is created instead of improving the existing one. Although this approach makes the program bigger in size, it lowers the execution time and increases the maintainability by achieving minimal coupling and high cohesion within the system components [Sommerville 1996b and Constantine 1979].

4.1.3.1 Validation of bibSearch.pl

The program *bibSearch.pl* is implemented and tested in a step-wise manner. After the implementation of the overhead computation, program modules for each form field is coded separately one by one. This approach provides the opportunity to test and validate the program easily and efficiently. After the implementation of each field module, first the functionality of the field itself independent from any other field is completely tested and the bugs are fixed. The test cases are recorded, which are used again when new field modules are added to the program. Second, the previously implemented fields are tested again with some of their test cases to assure that the new added field did not cause any change in their execution. Third, as final step, the newly added field is tested with the combination of the other fields, the results are validated, and the detected bugs are fixed. This procedure is applied field by field till the implementation and also the validation of the entire program is complete.

The correctness of the test case results is validated by composing the same search queries in FileMaker® Pro[Claris 1992], the database management system used for maintaining the bibliography databases, and comparing the results of the DBMS with the results of WebBiblio's search engine. As long as it is possible to compose the same query by using the search capabilities of FileMaker® Pro, this approach is useful. To determine the expected test case results for the advanced queries of WebBiblio, the search engine of FileMaker® Pro is used in a step-wise fashion. First WebBiblio's advanced query subdivided into simpler queries that can be run by FileMaker® Pro's search engine. Then

the results of these simple queries are combined to obtain the test case result for WebBiblio's advanced search query.

WebBiblio users have been encouraged to report bugs as well as their comments to WebBiblio developers. The initial page of the *Help* utility displays the sub-topic "Comments, bug reports". Selecting this sub-topic activates the e-mail application of the user's environment and enables the user to send comments or bug reports to the developers. So far, no bug report has been received.

4.1.4 The Abstract-Keywords Engine abskw.pl

The PERL program *abskw.pl* is executed when an abstract and/or keywords request is received. As also shown in Figure 4.4, the results of the search queries are listed with the buttons that can be used to view the abstract and/or keywords of the publications cited by the selected references in the result list. If no button is attached to the reference information, for instance third reference in the figure, then that means the keywords and abstract of the source cited by that reference is not available. Among the three buttons, the leftmost one can be used to view the abstract, the middle one the keywords and the rightmost one both: keywords and abstract of the cited source. When one of these buttons is clicked, the user's request is handled by the abstract-keywords engine of WebBiblio: *abskw.pl*.

The engine receives one input value from the web page: the identification name of the button. The name contains the necessary information for the program to determine user's request: the type of button and the reference which the button is attached to. By

processing the input value, the request is analyzed and the information is retrieved in a direct, efficient way by using the related data files, which are explained in Section 4.2.3 on page 68 and in Section 4.2.4 on page 70. As the final step, *abskw.pl* displays the retrieved information with the reference and main menu of WebBiblio on a new web page. Figure 4.5 shows the display of abstract and keywords information of the reference's source by *abskw.pl* upon the user's request by clicking rightmost (both) button attached to the fourth reference on the web page shown in Figure 4.4.

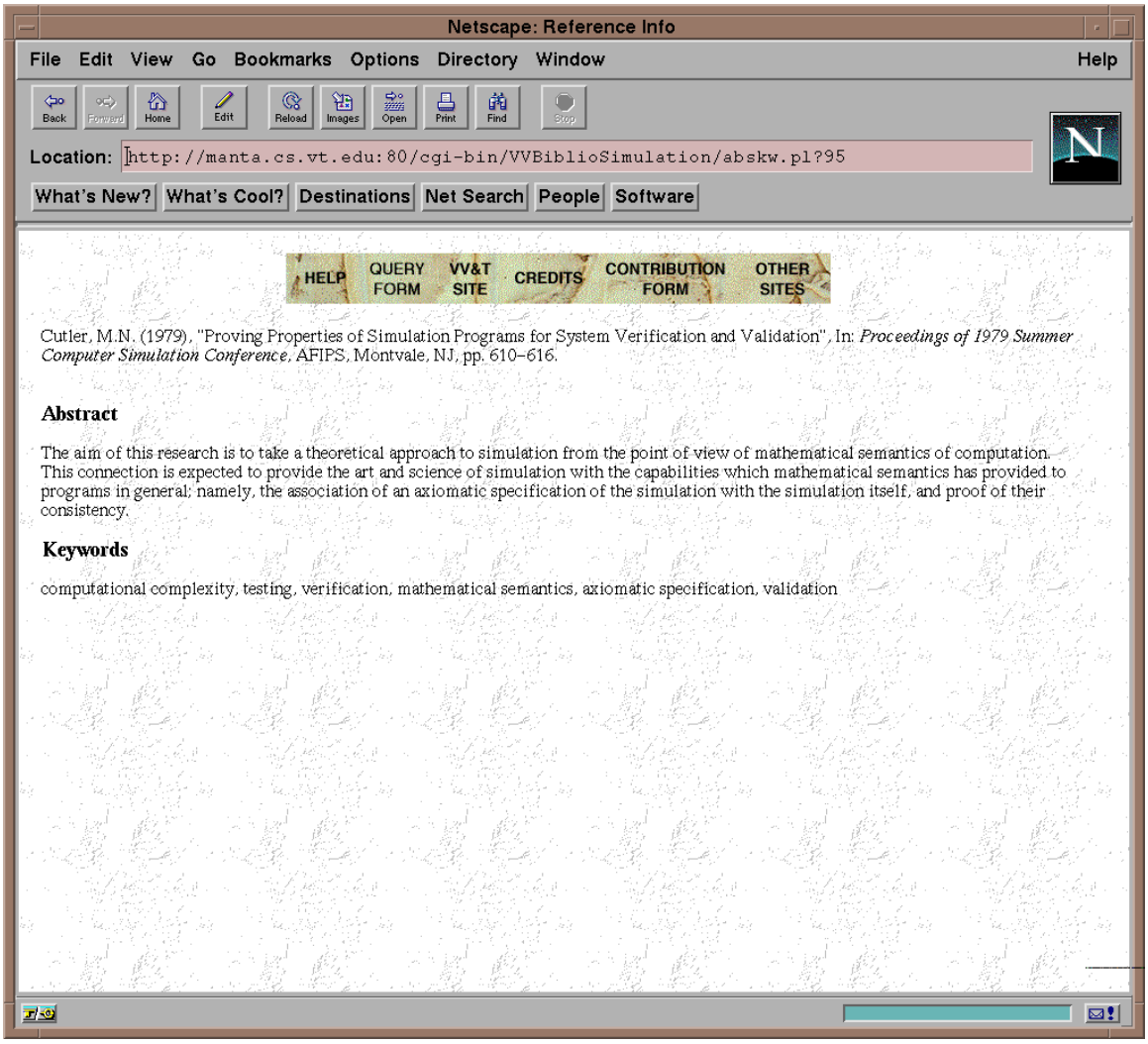

Figure 45 Display of a reference's abstract and keywords by *abskw.pl*

4.2 Data Files

Based on Objective 8 of the study, WebBiblio is designed and implemented in order to provide the fastest response time possible. Hence, use of a DBMS in the background to answer user's search queries would conflict with the objective of the study. This approach would cause additional time in responding the users queries because of the time spent on communication, and also query and result conversion between the query parser and

DBMS. Instead of this approach, by designing special data files that contain the bibliography database information in a special format, query and result conversion is eliminated as well as communication and searching time is minimized.

Hence, WebBiblio publishes each bibliography by using 64 data files that are created from their corresponding bibliography database. The data files vary in their structures and contents to minimize the response time. Figure 4.6 summarizes the classification of the data files according to their structures, contents and usage with the list of data files belonging to each class. The following sections explain the characteristics of each data file class.

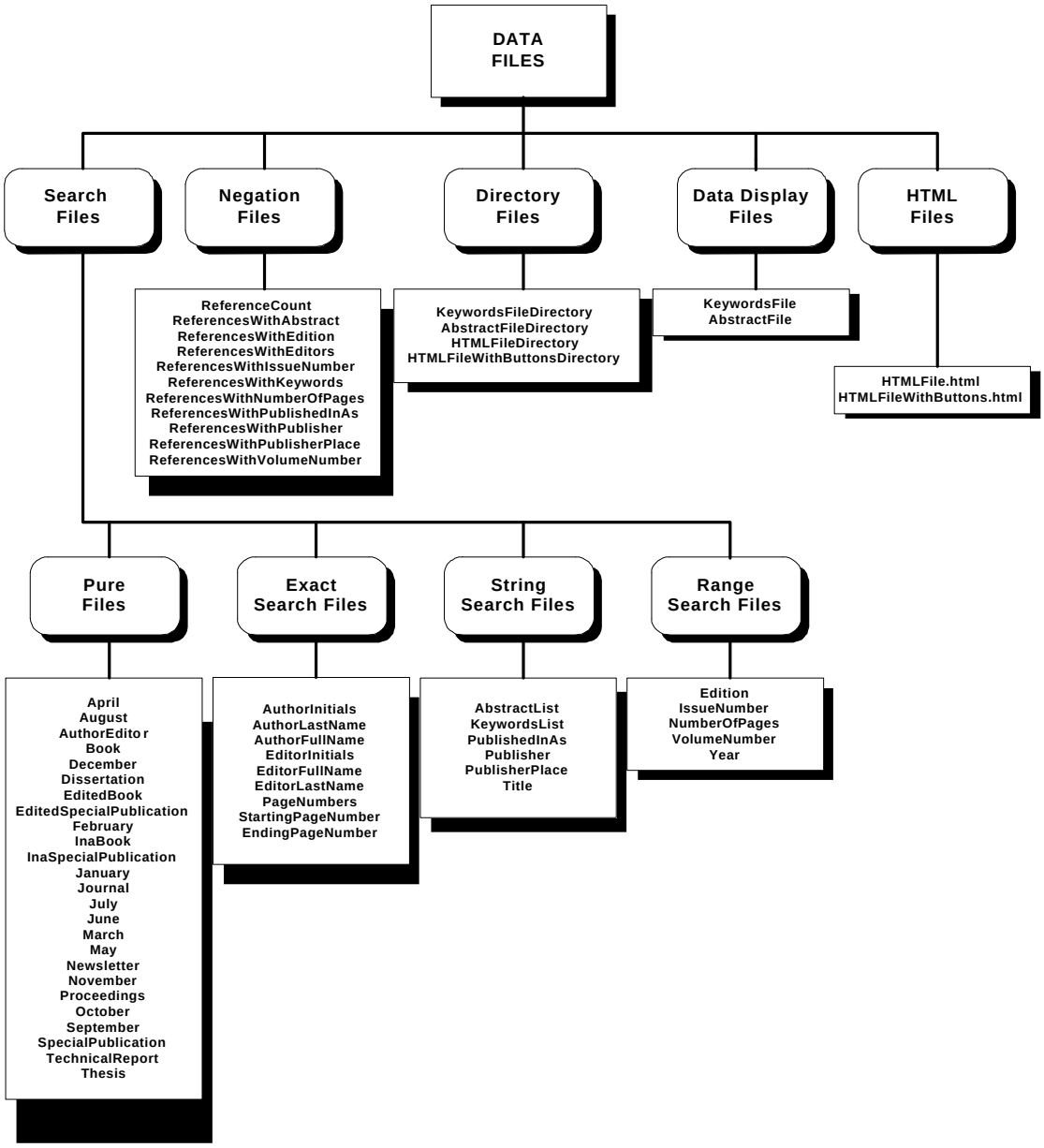

Figure 46 Classification of Data Files

4.2.1 Search Files

Every reference in the bibliography database has a unique number that is basically used as the reference's identification number. The search operation is performed over these

reference identification numbers. After the reference numbers that match to the query are determined, then the actual reference information is retrieved based on these numbers to display the search results.

Search Files are used to determine the identification numbers of references that match to the search query. Most of the data files belong to this class. According to their structures and the searching process applied on, they are divided further into four subclasses: Pure Files, Exact Search Files, String Search Files and Range Search Files.

4.2.1.1 Pure Files

These files contain only reference numbers. Practically no search is made on these files, since all references whose numbers are in these files match to the search query. Basically the search engine determines the name of the data file and after that all the references whose numbers are in that file match to the query.

This approach is applicable for the fields where the set of all possible queries is well defined. These fields are the list menu fields *Publication Type* and *Month,* and the checkbox field *Editor*. These are the fields, where the user does not type a query but makes a selection from the menu, or among the two values in checkbox case. Hence, the set of all possible queries that can be composed using these fields are known, and the results of these queries can be pre-prepared.

Let us consider *Month* field. The queries composed by using this field can be only a subset of the month names set. For sake of simplicity, let us assume the query is *August*. The data file *August* simply contains the numbers of references that refer to the sources

published in August. Hence, the answer to the query is all the reference numbers in this data file. The search is pre-made, all the search engine needs to do is to read the numbers from the file, which are also already sorted. If the query consists of more than one month name, the process is followed for every month in the query, and the results are combined by using logical OR-operation, which is explained in Section 4.1.2.1 on page 37. The searching process is same for the publication types.

Similarly, *Editor* checkbox has only two possible values, *checked* and *unchecked*. The default value *unchecked* has no effect to the search. When the box is checked, then the numbers of references matching to this query automatically obtained from the data file called *AuthorEditor* without any search operation.

To conclude, 12 publication type files, 12 month files and the data file *AuthorEditor*, in total 25 of data files are Pure Files, which enable the search engine answer the queries of *Publication Type*, *Month* and *Editor* fields without searching.

4.2.1.2 Exact Search Files

Each line of these files contains a unique key value and a non-empty set of sorted reference numbers. The key and the numbers are separated with pipe character (|), and the numbers with comma (,). A hit occurs, if the search string exactly matches with one of the key values. Then the references whose numbers are listed in the line of the hit key match to the search query. For instance, if the search string is Jones and the data file contains a line like Jones|3,78,79,80 then the references identified by these four numbers match to the query. Since the key values in the files are unique, a hit terminates the search.

The search of input string among the key values is sequential. Since the sizes of lines in the file vary, binary search is inefficient and not trivial. Considering the size of the file, this straightforward algorithm is considerably efficient. For larger bibliographies, a hash table can be used to access the key directly in the file, since the time spent for extra computation and file I/O for the hash table can be compensated by the faster access to the searched key in the data file.

The data files used to answer queries composed by *Author Name*, *Editor Name*, *Page Numbers* fields, which are *AuthorInitials*, *AuthorLastName*, *AuthorFullName*, *EditorInitials*, *EditorLastName*, *EditorFullName*, *PageNumbers*, *StartingPageNumber* and *EndingPageNumber*, belong to this class.

4.2.1.3 String Search Files

Like Exact Search Files, each line of these files contains a unique key value and a non-empty set of sorted reference numbers. The key and the numbers are separated with pipe character (|), and the numbers with comma (,). A hit occurs, if a non-empty set of key values contains the search string (note that the same search operation applied on the Exact Search Files may be applied also on these files depending on the search query; in other words the user have the option to convert the string search into exact match search). Then the references whose numbers are listed in the lines of the keys that belong to the set match to the search query. For instance, if the search string is Los and the data file contains two lines like Los Angeles, CA|3,78,79,217 and Los Altos, CA|27,41 then the references referred by these six numbers match to the query. Each string search operation

is performed based on two flag values, case sensitivity and whole word match, which are explained in Section 4.1.2.5 on page 46. These flag values are also determined by the user and passed with the search query as part of the input.

String search over these data files is performed sequentially. Since the contents of all the keys in the file, in other words the entire data file, have to be scanned the sequential search with no overhead computation is the most time and space efficient searching algorithm for String Search Files.

According to the types of relations between the keys and the reference numbers in their reference number sets, these files are divided further into three subclasses as explained below.

In the first subclass, the keys and reference numbers have 1:1 relation; in other words each key has exactly one reference number in its reference number set. The data files *AbstractList* and *Title* belong to this subclass since abstracts and titles have 1:1 relation with the references (for instance, each reference refers to one abstract and each abstract belongs to one reference.) These files are sorted with respect to the reference numbers to reduce the response time. This design does not change the execution time of sequential search over the key values, but since the file is previously sorted with respect to the reference numbers, the matching references are detected in this sorted order because of the sequential reading of the file. Hence, after the search, the engine does not need to sort the numbers of the matching references for further computations, since they are obtained in sorted order.

In the second subclass, the keys and reference numbers have 1:M relation; in other words there can be more than one reference number in each key's reference number set, but each reference number appears only in one key's reference set. The data files *PublishedInAs*, *Publisher*, and *PublisherPlace* belong to this subclass. For instance, each reference has only one publisher, although a publisher may have many publications cited in the bibliography.

The reference numbers in keys' reference number sets are sorted, to speed a possible sorting at the end of the search. As a fact the improvement is asymptotical. For instance, the searched string is found in the contents of two different keys. Hence, the result is the sorted array that is the union of two disjoint reference number sets belonging to these keys. Since these sets are already sorted , the sorted union set is created by merging the arrays into one array, which takes O(n) time, where n is the number of the elements in the result array. Otherwise, concatenating two arrays to obtain the union set and sorting the elements in it with an efficient algorithm would take O(nlgn) time.

In the third subclass, the keys and reference numbers have N:M relation, which means there can be more than one reference number in each key's reference number set, and each reference number can appears in more than one keys' reference sets. The data file *KeywordsList* belongs to this subclass, since most of the publications cited by references have more than one keyword, and some keywords can be found in more than one publication cited in the bibliography. The file structure and the searching process for the data file in this subclass are exactly the same as the second subclass files. The only difference is that the sets that are merged into to obtain the sorted union set are not

necessarily disjoint, but this does not affect the algorithm except an extra conditional check during merging. Still O(n) time complexity for combining the search results is achieved.

4.2.1.4 Range Search Files

Like search files belonging to other classes, except Pure Files, each line of these files contains a unique key value and a non-empty set of sorted reference numbers. The key and the numbers are separated with pipe character (|), and the numbers with comma (,). All the keys in these data files are numbers as well. The data files belonging to this class are used to respond to search queries based on the number ranges. The input for the search operation consists of a set of the number intervals, and the result of the search is the union of reference number sets belonging to the keys that are covered by these intervals. For instance, if the input is the interval 1971-1973 and the data file contains the three lines 1971|23,45,67, 1972|1,8,533 and 1973|234,444 then the result set is the union set of these 8 numbers, which identify the matching references. The data files *Year, VolumeNumber*, *IssueNumber*, *NumberOfPages*, and *Edition*, i.e. all the data files corresponding to Numerical Query Fields, belong to this class.

The input intervals are previously sorted and organized to prevent any overlap and redundancy so as to decrease response time. These sorted intervals are sequentially searched over the key values, according to which also the lines of the data files are sorted. Hence, for the entire set of the intervals the corresponding data file is scanned only once.

In most cases, the file is not scanned entirely, since the search terminates after the last interval is processed.

In Range Search Files, the keys and the reference numbers have 1:M relation; in other words there can be more than one reference number in each key's reference number set, but each reference number appears only in one key's reference set. For instance, each reference has only one publication year, although many publications cited in the bibliography may have the same publication year. Hence, the same algorithm is used to unite the reference numbers sets of matching key as the algorithm used for the String Search Files having 1:M relation among the keys and reference numbers. Consequently, the result set is created with O(n) time complexity instead of O(nlgn) after the sets to be united are determined.

4.2.2 Negation Files

These data files are used for efficient negation operation. Except the data file *ReferenceCount*, all of them have the same structure as the files in Pure Files subclass; they only contain comma separated reference numbers. *ReferenceCount* contains only one integer indicating the number of references in the bibliography database.

These files are used in the determination of the universal set for the negation operation. For the fields, *Author*, *Title*, and *Year*, since all the references in the database have a data entry, the universal set for the negation operation on these fields is all the references in the database. Hence, the file *ReferenceCount* is used for the negation operations on these fields. Negation of input is computed by excluding the integers in the

input array from array containing all the in the integers from 1 to number of references, which is retrieved from *ReferenceCount* data file.

For the negation operation over the fields, in which the references in the bibliography may have an empty entry, corresponding negation files are used. For instance the data file *ReferencesWithAbstract* contains the numbers of all references in the database whose abstracts are available. This file is used for the negation operations performed for the queries of *Abstract* field. Let us assume that the user wants to see all the references whose abstracts do not contain the word accreditation. To answer this query, first the reference entries in the database whose abstracts containing this word are determined. This is a subset of the universal set that is created by using the file *ReferencesWithAbstract*, which contains the numbers of all the references with abstract entry. By extracting the numbers in the result subset from the universal set, the negation operation is performed.

Similarly, nine more negation files are used for the computations of negation operations in the queries of *Published In/As*, *Volume Number*, *Issue Number*, *Publisher Name*, *Publisher Place*, *Editor Name*, *Edition*, *Number of Pages* and *Keywords* fields.

4.2.3 Directory Files

Directory files are used to determine the location of the requested information in data files. By using this location information, the data is accessed directly from the file without any search, so that response time is minimized in retrieving the requested information and displaying the search results. Each directory file is associated with an *HTML File* or *Data*

Display File, for which it provides the directory service. The data files *KeywordsFileDirectory*, *AbstractFileDirectory*, *HTMLFileWithButtonsDirectory* and *HTMLFileDirectory* are directory files of the data files *KeywordsFile*, *AbstractFile*, *HTMLFileWithButtons.html*and *HTMLFile.html*, respectively.

The directory files can be considered as data access tables for their associated data files. Each line of the files contains a reference number, the location of the beginning of the related information in the associated data file and the length of the information in number of characters. The reference number and the location are separated by pipe character (|), and the location and length by comma (,).

The type of the related information depends on the directory file. If it is *AbstractFileDirectory*, then this information is the abstract of the publication cited by a reference. Similarly, it is the keywords information or the reference itself in HTML [Graham 1996] format when the other directory files are considered.

The processing on these files is simple. The engine basically looks for the table entry of the reference in the file about which the information is requested. It makes a sequential string search for the input reference number over the file by using *grep* function of PERL [Quigley 1995], and returns the location and the length values associated with this reference number. The pipe character (|) appended to the search string before the function is called, so that the digits of location or length values in the file cannot be considered as a match with the search string. Since only reference numbers in the file are followed with pipe (|) character, an incorrect hit does not happen. When the size of directory files (approximately 12 bytes per record in the bibliography database) and the

simplicity of the algorithm are considered, the sequential search using efficient *grep* function seems as the best option for the execution efficiency and maintainability.

4.2.4 Data Display Files

These data files contain only information without any additional structure. Only two data files belong to this class: *KeywordsFile* and *AbstractFile*. They contain the keywords and abstracts of the references entries in the bibliography database, respectively. The information in these files is accessed with their associated directory files. By using the information in the directory files, the abstract and keywords of a specific reference can be directly retrieved from these files. Since the beginning location and the length of the data are known in advance with the help of the associated directory files, no search operation is performed on this files to retrieve the information. When considering the large sizes of the data display files, direct access to these files without any search plays the key role in achieving the fastest response time.

4.2.5 HTML Files

The data files belonging to this class contain only information without any additional structure similar to the Data Display Files. The difference is that the information is provided in HTML [Graham 1996] format so that the retrieved data from these files can be displayed on a web page without any further processing. Two data files belong to this class: *HTMLFile.html and HTMLFileWithButtons.html*. Both of the files contain the references of the bibliography database in HTML format. The difference is that the second

file also contains the clickable images, the colored spheres, which can be used to view the abstracts and/or the keywords of the references.

HTMLFileWithButtons.html is used to display the search results with clickable buttons. On the other hand, *HTMLFile.html* is used to display the corresponding reference information with the display of requested abstract and/or keywords information. Since for this case, these clickable buttons are unnecessary, this second file containing the reference information in HTML format without images is required.

Like the data display files, HTML files have associated directory files. By using the information in these directory files, the beginning location and the length of the requested information in the HTML files are obtained, and the reference information is directly retrieved from that location without any search. Again the large size of these files makes the direct access essential for them to minimize the response time.

4.3 Bibliography Databases

The bibliography databases are collections of related references for each published bibliography including the available abstracts and keywords of sources cited by these references. They are created and managed by using FileMaker® Pro[Claris 1992] database management system. All the changes in the databases - updates, contributions, corrections - are first made by using FileMaker® Pro. Then, the changes are propagated to the database structure of the search engine by creating the data files from the bibliography databases in a HyperCard system using HyperTalk 2.0 [Winkler 1990] scripting language.

As also previously mentioned, publication of bibliographies by WebBiblio is independent of the DBMS used. Any DBMS software can be used as long as the data files are created from the database according to their specified formats.

4.3.1 The Creation of the Bibliography Database

To fulfill Objective 12 of the study, a Simulation VV&T Bibliography database is created and developed to be published by WebBiblio when it becomes available on the WWW. The bibliography database is also used in verification, validation and testing of WebBiblio throughout its development.. This section describes the creation and development of this bibliography database.

This was the most time consuming process of the study. To have an important contribution and to assure user satisfaction, besides all the other provided features, creation and publication of a rich bibliography database also containing most recent references from a wide range of sources is aimed. To achieve this goal, throughout the entire study, the improvement of the database has continued in parallel.

The core of the database is created from the Dr. Balci's collection of the related publications by entering the bibliographic reference data of publications into the database. This collection, which mostly contains references published in 70s and early 80s, composes the seed of the bibliography. By adding the reference data of recent publications into the database, the bibliography is improved and made up-to-date.

These publications are searched and found by scanning all the proceedings of the related conferences, like Summer Computer Simulation Conference and Winter Computer

Simulation Conference, and all the related journals published between 1980 and today. Following that an intensive literature survey on books, special publications, technical reports, newsletters, dissertations and theses for the related publications is made, and the bibliographic reference information of found ones is entered into the database. The contributions of the contributors become another important source of the publications for the bibliography database. Approximately 5% percent of the bibliographic reference information in the database is obtained as contributions.

Entering the bibliographic reference data of a publication into the bibliography database does not complete its import process. Based on Objective 7 and Objective 12, the abstract and keywords of the publications are also required information to be collected. The procedure below is followed for obtaining and entering abstract and keywords information of the publications into the database.

First the abstracts and keywords of the publications are searched and found. This is done with an intensive library and WWW search. The abstracts obtained as hard copies are photocopied for the second step.

As the second step, the abstracts are entered into the database. The ones found through the WWW are imported easily by using “copy and paste” method. For the abstracts retrieved as hard copies, OmniPage® Pro [Caere 1996], an OCR (Optical Character Recognition) software is used. The time spent on entering the abstracts on printed material is significantly reduced by the usage of this software instead of typing the data. The high speed and accuracy rate of the software play an important role in reducing processing time of this step. Although it mostly depends on the quality and cleanness of

the printed material's paper and the font of the original script, for most cases the software finishes the process with more than 95 percent accuracy rate.

After the application OCR operation, the soft copies of abstracts are obtained. On these copies, the words found as suspicious by OmniPage® Pro are corrected and complete spell checking of the abstracts are made. At this point, an acceptable accuracy rate for the import of the abstracts is achieved. The spelling mistakes that also exist in the original script are not corrected to preserve the originality. In case of the contributed abstracts, the abstracts received are considered same as the originals, hence they are added into the database without any change.

As the final step, the keywords of the publication are entered into the bibliography database. If the keywords are provided by the author(s), they are simply typed, or copied and pasted if they are obtained as soft copies. In most of the publications the author(s) omit providing the keywords. For these cases, the keywords are determined by focusing on the titles, abstracts and the section headers of the publications. When determining the keywords, the keyword *simulation* is omitted, since every publication in the bibliography is expected to have this keyword.

4.4 Help

A detailed on-line help utility is provided for all features of WebBiblio so as to meet all possible needs of WebBiblio users. To achieve this goal, a detailed, simple, easy-to-use, consistent *Help* utility with a rich set of examples is written as part of WebBiblio.

Help contains guidelines even for the most trivial actions for any part of WebBiblio. For each of these parts an introductory description is written, and the more detailed information is added under the titles following this initial section. Each individual field of *Query Form* is explained separately in detail. For each field, a precise description, informative examples and its query composition syntax (if applicable) are given. Likewise, each field of *Contribution Form* is explained individually in detail. Screen-shots containing the images of filled contribution forms are included as examples to guide contributors. For both of the forms, in the explanation of *Publication Type* field, one example from each publication type in the bibliography database is included to guide the users in determining the publication type of the reference that they are searching or contributing.

Help is provided through a user friendly interface. It is displayed by using the frame structure of HTML [Graham 1996]. The top frame and bottom frame contain the main menu of the bibliography and the main menu of *Help*, respectively. Consistent with the main menu of the bibliography, the main menu of *Help* is also a clickable image. The middle frame is also divided into two horizontal frames. When a main topic from *Help*'s main menu is selected then the right frame lists the sub-topics under this topic, and the left frame displays the contents of *Help* for the selected sub-topic from the right frame. Initially, the left frame displays the contents for the first sub-topic in the list of the right frame. The hierarchical structure of Help contents is summarized in Figure 4.7.

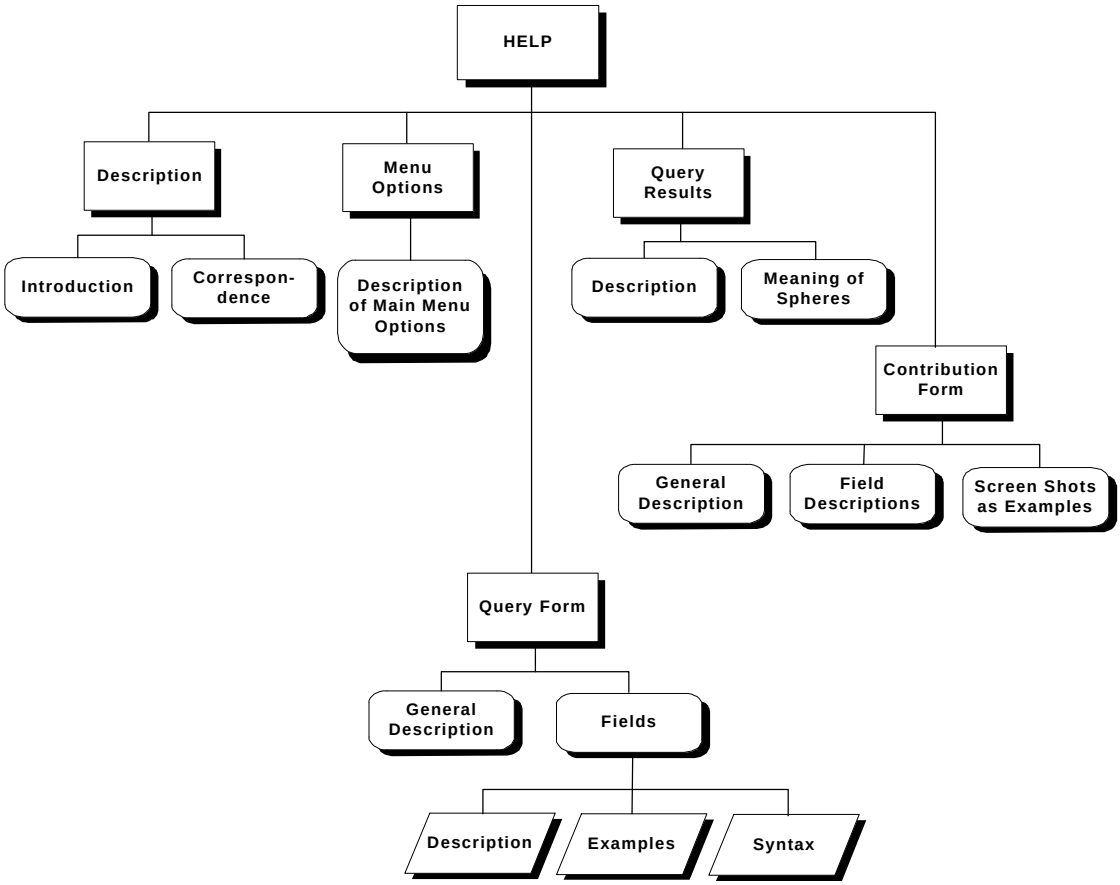

Figure 4.7 The Structure of *Help*

When *Help* is launched, the main topic *Description* is selected and the left frame displays the contents of its first sub-topic *Introduction*. *Introduction* provides a general description of the bibliography and the basic guidelines to use the *Help* utility as depicted in Figure 4.8.

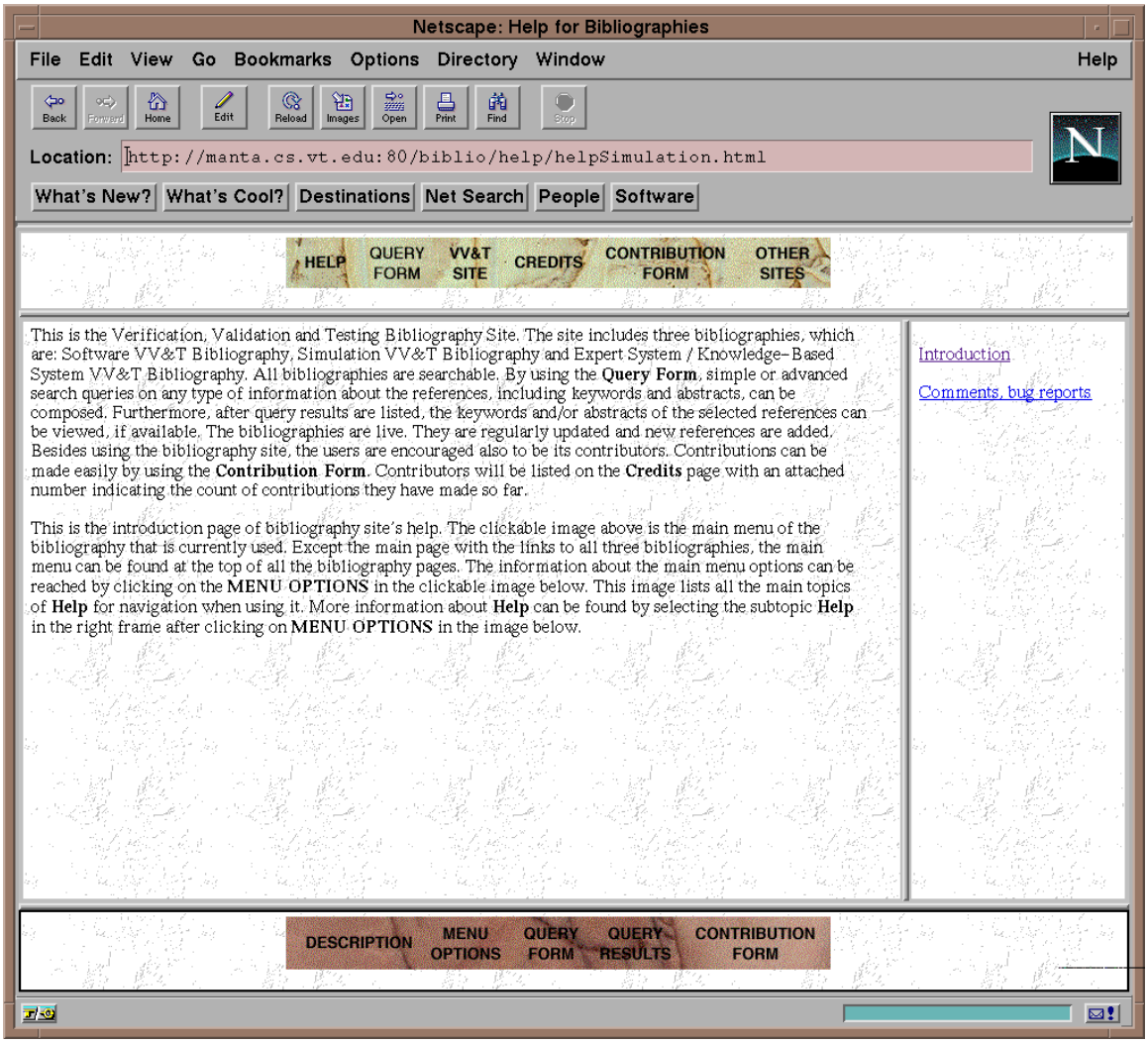

Figure 48 WebBiblio*Help*

The help information is provided in a consistent manner. Similar elements of WebBiblio have the similar *Help* contents explained in the same structure. For instance, all fields of *Query Form* have examples in their descriptions consistently, and also these examples are selected in a consistent manner so that the fields that have relatively closer usage and syntax have similar examples to pinpoint the similarities and also the differences between these closely related fields.

Hierarchical structure and consistency of *Help* simplifies the navigation and the understanding of the contents, and consequently reduces the time spent by the user to find and understand the answers for his/her questions when using the software tool. Hence, the usability of the software product is increased by improving the learnability with the reduced learning time [Sommerville 1996a].

4.5 Credits Page

WebBiblio contains a separate page for credits for each published bibliography. This page displays the developers of the software system and contributors of the bibliography. Both of the lists are presented in bordered HTML tables [Graham 1996]. The users of WebBiblio are also encouraged to be its contributors. User contributions are essential to enrich the bibliography databases and make it up-to-date, and also to correct any inaccurate information that they may contain.

To encourage contributions, all contributors with minimum ten contributions are listed on the *Credits* page. Number of contributions is attached to each contributor's name. The list is sorted in descending order with respect to these contribution numbers (in case of tie, sorting is carried out with respect to last name.) The *Credits* page of the Simulation VV&T bibliography is shown in Figure 4.9.

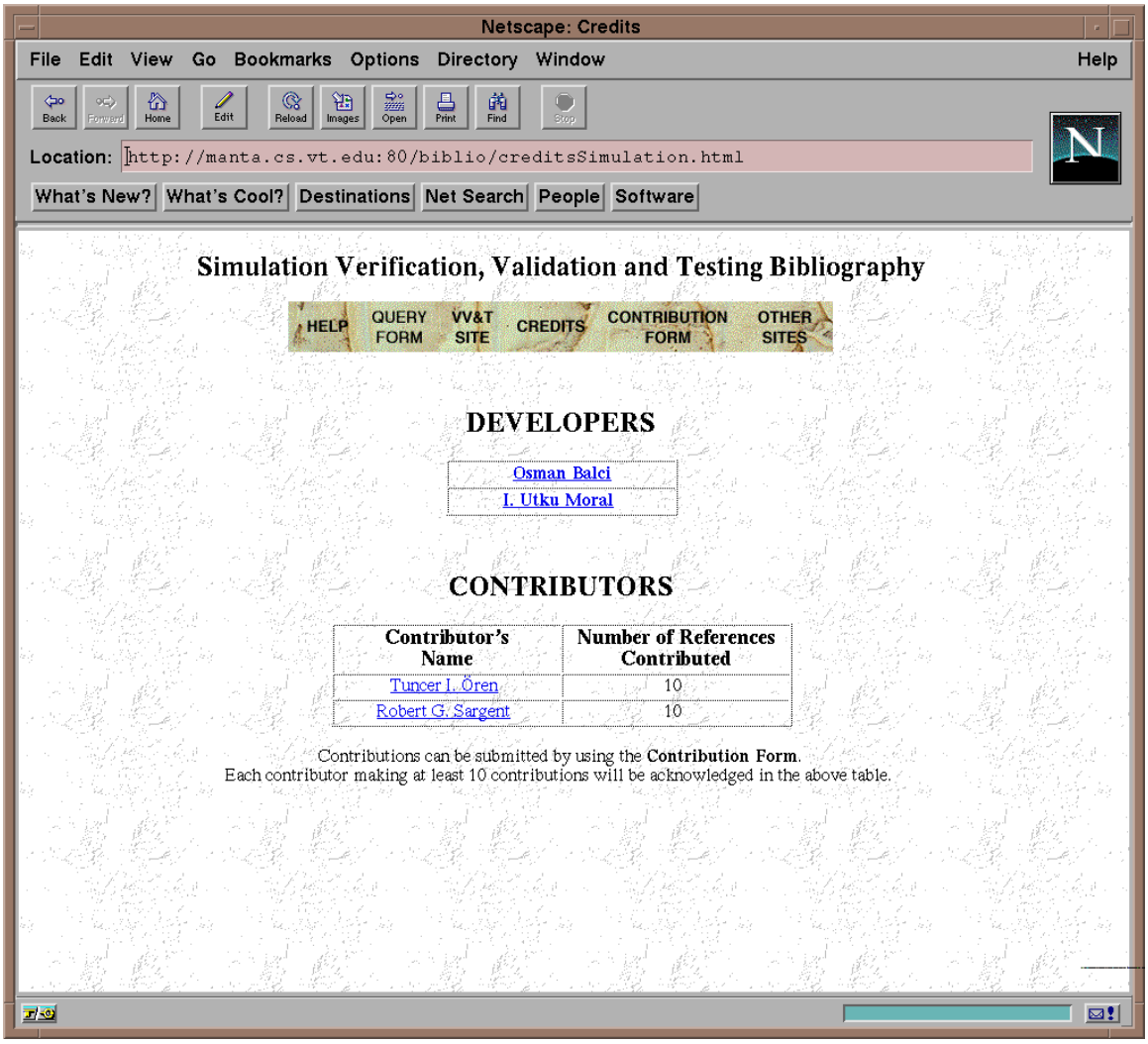

Figure 49 *Credits* Page

4.6 Other Sites Page

This page of each published bibliography contains links to other related sites that may be of interest to users of the bibliography. Users of these linked sites can be also the potential users of the bibliography published by WebBiblio, which may make the developers of these sites include links to WebBiblio at their web pages. Consequently, by including this

page for each published bibliography WebBiblio contributes to the inter-connection of related sites; so they are ‘webbed’ together.

4.7 Contribution Form

WebBiblio provides a contribution form for each published bibliography to give the users the opportunity to contribute to the published bibliographies. For the sake of consistency and simplicity, the fields of the form match exactly to the fields of reference entries in the database with the same order. It has also the same field order with *Query Form*, although these two forms are slightly different since their fields are not identical. For instance, *Contribution Form* contains URL fields to get the necessary information for WWW links when displaying the search results, and contributor related fields for authentication, listing in *Credits* page and possible correspondence. Figure 4.10 shows *Contribution Form* launched under the Simulation VV&T bibliography.

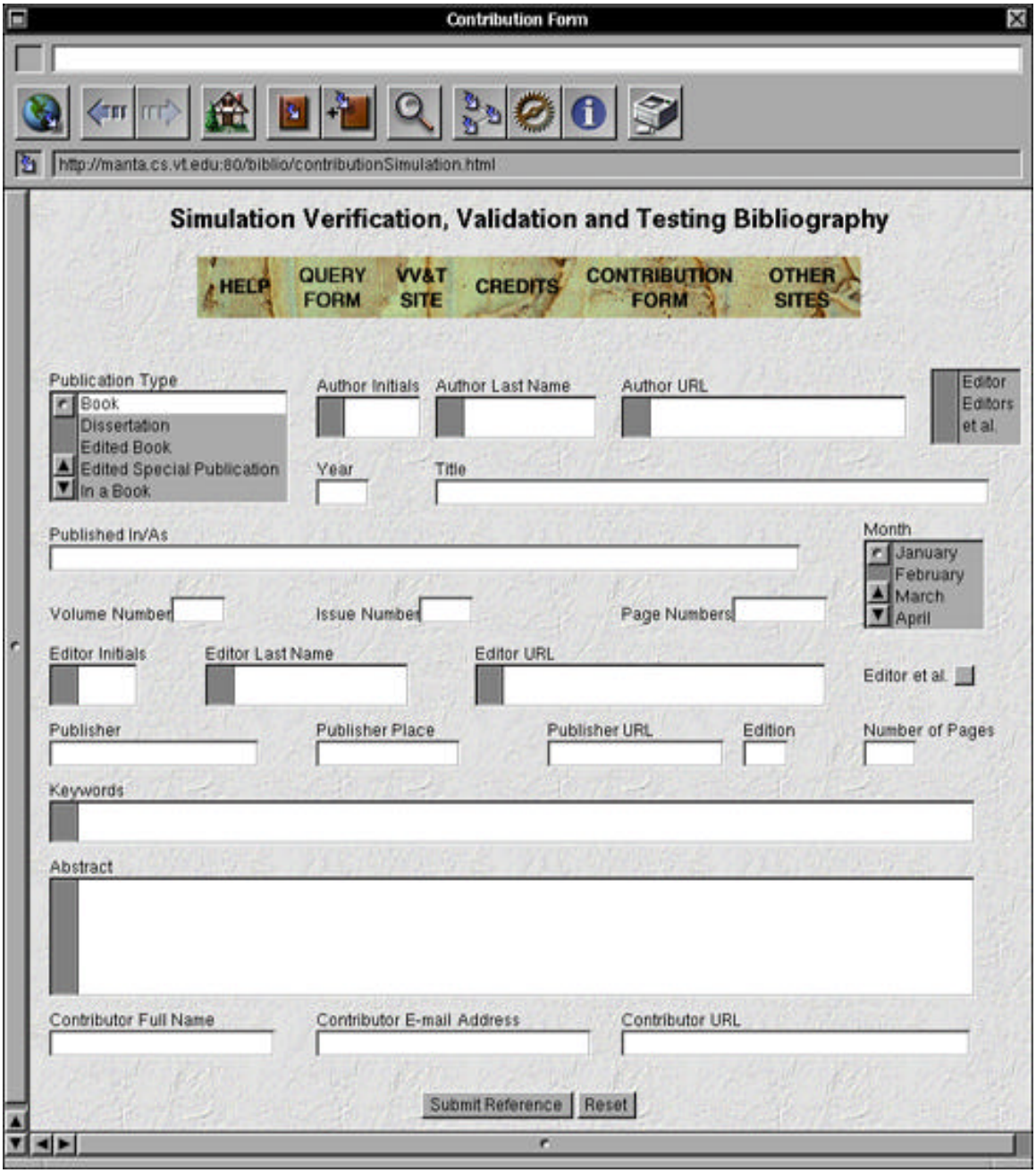

Figure 4.10 WebBiblio*Contribution Form*

Each contribution is sent as an e-mail to a special dummy user account on the server, which is under the name *biblio*. With certain periods, the contributions, in other words e-mails of *biblio*, are processed and the bibliography databases are updated with the

import of contributions. This routine administrative task is a simple, three-step procedure, which is explained specifically in Section 5.1 on page 87. All the remaining details of the conversion and importing process of contributions are handled by the PERL program *contribute.pl* that runs when the *Contribute Reference* button of the *Contribution Form* is pressed.

4.7.1 The Contribution Engine: contribute.pl

The PERL program *contribute.pl* has two basic duties: 1) Conversion of the user's contribution, the input of the program, into an importable format for the bibliography database, and 2) Sending the data in this format as an e-mail message to *biblio@manta.cs.vt.edu*

The conversion process can be subdivided into four steps: 1) Validity check of the contribution, 2) Data conversion, 3) Operating system conversion, and finally 4) DBMS format conversion.

As the first step, unfortunately, the program cannot make a complete validity check of the contributed information because of the nature of the situation. It is mostly the contributor's responsibility to provide accurate information. For instance, the contributed abstracts are always considered accurate, spelling mistake free. Although a spelling check can be performed, the contributed abstracts are considered the original and imported into the database without any change. This approach is followed not only for the abstracts but also for all the information in the contribution. To achieve the accuracy in the contributions, each field of *Contribution Form* is explained in *Help* in detail as guidelines

for the contributors. The syntax and the style of each field are clearly defined. Besides this detailed explanations, some screen-shots containing examples of filled forms are also provided to guide the contributors. These example contributions are carefully selected, so that each of them contains contribution of a reference belonging to a different publication type and using different set of form fields.

Under these considerations, the basic validity check of the contributions consists of checking the form fields that are required to be non empty. These fields are *Author Initials*, *Author Last Name*, *Year*, *Title* and *Contributor's Full Name*. If at least one of these fields in the contribution is empty then the contribution is not accepted, the submission is blocked, and the contributor is warned with an informative message. Every reference in the bibliography database has non-empty publication type, author name, year and title entry. Hence any contribution missing at least one of these information is not acceptable to preserve the consistency of the database (as a matter of fact, all contributions have automatically a *Publication Type* entry, since there is always a default selection, *Book*, at the *Publication Type* list menu of the *Contribution Form*. Consequently a contribution without *Publication Type* is not possible.) On the other hand, contributor's full name is also required for authentication purposes, and to be able to list the contributor's name on the *Credits* page.

Besides confirming the presence of information in required fields, if the contribution contains recoverable invalidities, these are corrected by the program. Since these are recoverable, the correction is done by the program without notifying the contributor, and the submission is accepted. Author or editor name initials in lower case

letters instead of upper case, space characters between these initials are examples of recoverable invalidities.

In second step, the program makes the necessary data conversion. Since publication type names and month names are represented on *Contribution Form* and bibliography database differently, this step is essential. In third step, the operating system conversion is performed by replacing two character end-of-lines of NEXTSTEP with one character carriage-returns of Macintosh.

As the final step of the data conversion, the program combines the information in the form fields by using the tab character as separator between them, so that later on the information can be imported by FileMaker® Pro [Claris 1992] properly as tab-separated record.

Tab-separated format is preferred rather than comma or carriage-return separated format since these two later characters can be used in the fields of the form and they do not have alternatives. On the contrary, the usage of tab character is not essential, it can be always replaced by space character(s). As a matter of fact, the contributor should not use tab characters when filling the fields of the form, since none of the field entries of the references in the database has more than one space character between any two words; but still the program has to consider the possibility that the contribution may contain tab characters. This is handled as one of the recoverable validity violations and tab characters are converted into one space character before the contributed information is combined into tab-separated format. The third and fourth steps explained above depend on the selected DBMS and its operating system and may vary according to the software decisions of the

bibliography publishers using WebBiblio. For sake of consistency, these steps are explained based on the selected DBMS and its operating system that are used in the development of WebBiblio.

Finally, the program sends the contribution in its formatted compact form as the body of an e-mail message to *biblio@manta.cs.vt.edu.*

The PERL program terminates by creating a WWW page containing the main menu image of the bibliography and an appropriate message. If the contribution was invalid, the message informs the contributor and explains the problem. In case of valid contribution, it notifies the contributor about the successful submission and thanks to her/him for the contribution.

4.7.1.1 Validation of contribute.pl

Validation of the *contribute.pl* is done with the experimental usage of *Contribution Form*. As the first step, the program is tested with invalid contributions. Each of the five information required form fields are tested separately by submitting the contribution with empty required field and the appropriate error message is confirmed. Following that, tests with the contributions having more than one empty required fields are performed to complete this step of validation.

Secondly, contributions with recoverable invalidities are submitted and the received contribution e-mails are checked to confirm the proper execution of the program and the recovery of the invalidities.

Finally, valid sample contributions are made to test the proper import of the contributions into the bibliography database. These contribution e-mails are collected, processed and imported into the experimental duplicate of a bibliography database to confirm accurate modification of the database with the contributions. This step is repeated with different type of contributions and different number of contributions. The imported references are examined and the correct placement of the contributed information into the related database entry fields is confirmed.

CHAPTER 5 : WebBiblio ADMINISTRATOR'S MANUAL

This chapter describes the regular tasks of a WebBiblio administrator in detail. Based on Objective 9, these tasks are simplified as much as possible to increase the maintainability of the system. All the administrative tasks consist of a-few-step procedures, and none of these steps requires deep knowledge of the system components, in spite of the different operating systems used in the system.

The administrative procedures described in this chapter are based on also the DBMS selected for this research to maintain the bibliography databases published by WebBiblio. As previously mentioned, WebBiblio is independent of DBMS used, and any other DBMS can be used for the maintenance of bibliography databases as well. By describing the administrative tasks specifically in this chapter, we believe that some guidance is also provided for WebBiblio administrators preferring other DBMS's for the bibliography database maintenance.

The system administrator has three tasks that must be done regularly: (1) importing the contributions into the database, (2) updating the data files based on the updated DBMS database, and (3) updating the *Credits* and *Other Sites* pages of WebBiblio, when necessary. These three tasks are explained in the following sections in detail. The first two are described step by step.

5.1 Importing the Contributions into the Bibliography Database

Interested people can submit new bibliographic information to WebBiblio system administrator for publication on the web. *Contribution Form*, shown also in Figure 4.10 on page 81, can be used for these submissions. The contributions are made by entering the information about the contributed reference into the form fields and clicking on *Submit Reference* button of the form. Clicking on this button activates the contribution engine, the program running behind the form, which sends the entered information in a compact form to *biblio@manta.cs.vt.edu*as an e-mail message.

The contributions are imported by processing these e-mail messages. The administrator does not need to import the contributions one by one. Multiple contributions can be imported on regular time interval basis. The steps of importing the contributions by processing the e-mail messages are listed below:

- All the contribution e-mail messages are saved into a text file.
- The lines in the text file containing e-mail headers are deleted.
- After deleting the header lines, the text file is brought into the Macintosh environment containing the DBMS and the bibliography database (no further operation except deleting the header lines is necessary).
- The text file is opened by using the import menu command of FileMaker® Pro [Claris 1992], the DBMS used in this research, which is currently processing the bibliography database. When using the import command, the tab-separated fields option is selected.

After finishing these four simple steps, the database maintained by the DBMS is updated with the recent contributions. To update the bibliography published by

WebBiblio, the new data files must be created from the updated DBMS's database to replace the old ones. This process is explained in the following section.

5.2 Updating Data Files Based on the Updated DBMS Database

As explained in Section 4.2 on page 58, to minimize the response time, the search engine and abstract-keywords engine of WebBiblio operate on special data files created from the bibliography database maintained by the DBMS. The changes in a bibliography database are first made on the database maintained by the DBMS. Then they are propagated to the bibliography published by WebBiblio by creating new data files from the DBMS's database, and to replace old ones with them. The steps of this procedure are listed below:

- Records of the FileMaker® Pro [Claris 1992] database are exported into a tab-separated text file.
- The HyperCard software reads this text file and generates the 64 data files.
- The data files are imported into the directory *~root/httpd/cgi-bin/VV*Biblio/DataFiles* of the server machine where * stands for the name of the bibliography whose database is being currently updated (for instance the name of the directory for the Simulation VV&T is *VVSimulationBiblio*). The old data files in this directory are overwritten.
- The program *convertAll* is executed in the same directory to convert all the data files from Macintosh format into the NEXTSTEP format.

5.3 Updating *Credits* and *Other Sites* Pages of WebBiblio

Credits and *Other Sites* pages of WebBiblio may need to be updated by the system administrator when new contributions are made or new related sites on the WWW are discovered. To update these pages, the related HTML files in the directory */LocalLibrary/WebPages/biblio/* should be modified. The name of the files containing

Credits and *Other Sites* pages are credits*.html and otherSites*.html, respectively. Here the character * stands for the code name of the bibliography. For instance the name of the file for *Credits* page of the Software VV&T Bibliography is *creditsSoftware.html*.

On *Credits* page, two types of modifications are expected: change of a contributor's contribution count or addition of a new contributor. In the first case, the task is trivial. By overwriting the new count onto the old one it can be performed. To perform the second operation, the lines containing the HTML [Graham 1996] code for a contributor can be duplicated and the new contributor specific information can be overwritten on the duplicate copy. For *Other Sites* page, the expected update operation is the addition of a new site. This modification can be performed in a similar way to adding a new contributor into *Credits* page. By duplicating the lines of an already existing site and overwriting the new site specific information on the duplicate copy, the task can be completed.

CHAPTER 6 : EVALUATION OF WebBiblio

Evaluation of a developed software system is essential to assess the usefulness and effectiveness of its solution approach for the problem focused on in the conducted study. The evaluation of WebBiblio, a globally available tool, based on controlled experimentation is very difficult and beyond the scope of this study. Instead, a subjective evaluation of the software product is made.

This chapter provides this subjective evaluation with respect to the twelve research objectives stated in Section 1.2 on page 2.

Objective 1: WebBiblio should be a globally available tool on the WWW

WebBiblio is designed and implemented as a WWW software system. It is not a prototype. It is currently available on the WWW and publishing three VV&T Bibliographies. WebBiblio does not have any access restrictions to its web pages, hence it is a globally accessible tool by any WWW user.

Objective 2: WebBiblio should facilitate the publication of all possible elements of a bibliographic reference structure.

WebBiblio is designed and implemented to publish any type of scientific bibliography on the WWW. All possible elements of a bibliographic reference structure as well as abstract and keywords of the source cited by the reference are included. *Query Form* shown in Figure 4.1 on page 31 has 17 fields. We believe that these 17 fields cover most of the possible elements of a bibliographic structure.

Objective 3: WebBiblio should be well human engineered.

WebBiblio provides a user interface that satisfies the usability principles for interfaces [Nielsen 1990]: simple and natural dialogue, speak the user's language , minimize the user's memory load, consistency, provide feedback, provide clearly marked exits, provide shortcuts, good error messages, and prevent errors.

- **Simple and natural dialogue**

According to Nielsen [1990], a "simple" interface does not contain irrelevant or rarely needed information. Two forms of WebBiblio, *Query Form* and *Contribution Form*, have simple interfaces. They only contain form fields, field titles and the main menu of WebBiblio. No other information, like query syntax of fields, is provided although this information can be easily found using *Help*. Similarly, *Credits* and *Other Sites* pages provide only the related information with an easily readable presentation. *Help* utility provides detailed, but relevant information. With the use of HTML's frames [Graham 1996], the navigation is simplified. The language of *Help* is simple, natural and unambiguous. The explanations are supported by examples and screen-shots.

- **Speak the user's language**

The query syntax of *Query Form* fields is simple and consistent. There are no reserve words. User simply types qualifiers into the form fields as queries over the corresponding fields of reference information. The information provided in *Help* is clear, it does not require a specific knowledge to understand.

- **Minimize the user's memory load**

Complicated keystrokes are not required. All the actions are performed with single button clicks. Query composition syntax is simple. The set of special characters is small (six characters plus double quotation marks and parentheses) and they are consistently used within the syntax of different fields. All the web pages of WebBiblio, except *Contribution Form*, fits into a 17-inch screen (this also depends on the screen resolution). *Contribution Form* fits into a 17-inch screen vertically, and horizontally one mouse click is sufficient to see the rest of the form.

- **Consistency**

Consistency is effectively supported within every system component. All web pages of the system have the same plain background color as well as the main menu of the tool, which is located at the same place on the layout of each page. Every page contains the name of the bibliography that they belong to as header and all have meaningful page titles. Font sizes are used consistently in all pages. No colorful text is used.

Both forms are designed in a consistent manner and the layout of the fields follow the order of information presented in a bibliographic reference. The query syntax and use of special characters is consistent among the fields of *Query Form*. *Help* is also written in consistent manner. Contents for similar system components are presented with the same structure.

- **Provide feedback**

Error messages provide feedback to users to correct their search query. When a contribution is submitted, the contributor is notified and thanked if it is successful, or warned if a required information is missing in the contribution.

- **Provide clearly marked exits**

As being a WWW software system, the user can exit WebBiblio by simply downloading another web page. Besides, an exit to the web page containing links to the published bibliographies is provided as an option in the main menu of WebBiblio.

- **Provide shortcuts**

Publication Type and *Month* fields of the forms are designed as list menus to guide user and provide shortcuts in filling these form fields. Every bibliographic reference information is presented with links to the authors', editors', publishers' and publications' home pages, if they are available.

By allowing different formats, shortcuts in query composition are also provided. Allowing two digit year queries in *Year* field, number ranges in Numerical Query Fields, and to enter only initials in *Author Name* and *Editor Name* fields are some shortcut examples in query composition.

- **Good error messages**

When a syntax error is detected, the user is warned with an informative error message. The message contains the name of the field where the error is detected and the type of error. Similarly, contributors are notified if the contribution lacks a required

information. The message contains the name of the field that has to be filled for a successful contribution.

- **Prevent errors**

By providing simple syntax rules and explaining every component and feature in detail in Help, minimization of the user errors is aimed. Besides, the form fields whose values are well defined are designed as list menus to prevent syntax errors in these fields completely.

Objective 4: WebBiblio should provide an on-line assistance.

A detailed on-line help utility is created and provided as a part of the software system. It is structured in a hierarchical manner. Its hierarchy tree is represented with the usage of HTML's frames [Graham 1996] by displaying different levels of the tree within different frames. Hence, based on the hierarchical structure, the navigation among the contents of *Help* is simplified and speeded.

Every component and feature of WebBiblio is explained in *Help* in detail. Every field of *Query Form* and *Contribution Form* is described individually. For each field description, a verbal explanation, a specific decomposition of syntax rules and a reach set of examples with explanations are provided. Screen-shots are included to support the descriptions.

In designing *Help*, consistency is effectively supported to decrease the user's learning time. Hence, similar components of WebBiblio are explained with similar sentences, examples and structural decompositions. *Help* contents are written in a precise

and unambiguous language, which is easily understandable without deep knowledge of computer science terminology and English vocabulary.

To conclude, *Help* utility of WebBiblio with its features explained above fulfills the stated objective.

Objective 5: WebBiblio should provide advanced search capabilities over the published bibliographies.

Query Form of WebBiblio provides a wide range of search capabilities over the published bibliographies. By using *Query Form*, the user can compose simple as well as highly advanced search queries. With its 17 fields, it allows the user to compose queries over any field of a bibliographic reference structure, and also over the keywords and abstract information of the cited sources. With its simple and consistent syntax rules, the form is easy to use. For simple queries, the syntax is minimal. All the user needs to do is to type the search words into the related form fields. On the other hand, the user can enhance his/her queries further by including logical operators, parentheses and/or double quotation marks in the query, using multiple form fields simultaneously and changing the values of checkboxes. In other words, the user can make his/her query as advanced as s/he wants.

Hence, WebBiblio's *Query Form* is a powerful tool that fulfills this objective of the study.

Objective 6: WebBiblio should be verified, validated and tested as much as possible.

Two forms of WebBiblio as well as all other components of the system are verified, validated and tested thoroughly by using the Simulation VV&T Bibliography, which is developed in parallel. The forms are tested incrementally. After implementation of each

form field, first the functionality of the field is completely tested. Test cases containing invalid input are also applied to increase the robustness of the system and to verify the error messages. Then the functionality testing of the entire form is repeated with the integration of the new field. This procedure is repeated till the entire form is implemented and completely tested. During this process, to validate the search results provided by the search engine, the results FileMaker® Pro's search engine [Claris 1992], the DBMS used in this study, is used.

Besides this intensive verification, validation and testing of the system, by considering the global usage of WebBiblio, a link is provided in *Help* utility that enables the users to report bugs and write comments.

Objective 7: WebBiblio should provide the abstracts and keywords of the sources cited by the bibliographic references.

Besides the fields of a bibliographic reference, each reference entry in the bibliography database has also abstract and keywords fields to store abstract and keywords of the source cited by reference. Abstract and keywords information is provided to the user upon request. The buttons attached to the bibliographic reference information in the display of search results can be used to view this information.

Objective 8: WebBiblio should respond the user queries as fast as possible.

The search engine of WebBiblio is designed and implemented under this objective. It uses PERL [Quigley 1995] as the programming language, which supports time efficient character and string manipulation functions over the data files. Special purpose data files are created to improve the response time efficiency. With the usage of these files, some

search queries are pre-answered. For these queries the engine does not perform any search operation, but retrieves the pre-created answers from the relevant data files and displays the results. Similarly, no searching process is performed on huge data files. The information is retrieved directly by using the associated directory files that provides the location and the size of the requested information. On small size data files, the sequential search is employed, which outperforms the other search algorithms on small size files since it does not require overhead computation and use of special data structures. Based on data files' structure, the search query results are obtained in sorted order. Consequently, the results of different fields are combined by merging them with O(n) time complexity. To increase the time efficiency further, the function calls within the program are minimized, and the cohesion within the functions is increased.

Besides these, on the queries of Numerical Query Fields, redundancy detection is performed to clean the overlapping sub-queries. With this approach, redundant costly search operations are prevented by the cost of redundancy check operation.

Similarly, the abstract-keywords engine of WebBiblio is designed and implemented with consideration of minimizing the response time. The requested abstract and keyword information is retrieved from the corresponding huge data files directly with the help of associated directory files. Hence, the information is accessed by the program in most time efficient way without performing any search operation on the huge data files. Besides this approach, the input is simplified as much as possible to minimize the determination time of the users' request.

The third executed part of WebBiblio is the contribution engine. Although this program does not have any time critical processing, like performing search operations or data retrieval, it is still implemented in execution efficient manner to fulfill the objective.

In conclusion, by designing and implementing all the executed components of WebBiblio under these considerations, a high level of efficiency is achieved in its response time.

Objective 9: WebBiblio should be easy to administer.

The regular administrative tasks are simplified as much as possible. They consist of a-few-step procedures. All these steps are easy to perform. Besides, none of these steps require deep knowledge of system components in spite of the different environments used by WebBiblio.

Objective 10: WebBiblio should be easy to maintain.

All WebBiblio modules are well documented and structured to achieve high maintainability. In *bibSearch.pl* (the search engine), each field of *Query Form* is parsed and the related searching process is performed by a different module, where these parts are clearly distinguished by comments and sorted with respect to their appearance on the form. This feature provides high flexibility in modifying the logic or syntax of a field, adding new fields or deleting existing ones without any coupling with other modules.

High degree of maintainability in WebBiblio is achieved by high cohesion and low coupling among the modules [Sommerville 1996b and Constantine 1979]. The modules are coded in field specific manner, so that they are called by less number of modules that

handle different fields of *Query Form*. The communication between the functions is minimized resulting in low coupling.

Objective 11: WebBiblio should enable anyone to contribute to the published bibliographies.

By providing an easy-to-use contribution form, WebBiblio fulfills this objective. To guide the contributors, the usage of the form is explained in *Help* in detail with screen-shots containing sample filled contribution forms. To encourage contributors, the names of contributors with ten or more contribution are honored on *Credits* page.

Objective 12: A simulation verification, validation and testing bibliography database containing at least 500 references with abstracts and keywords should be developed, and WebBiblio should be used to publish it worldwide.

This bibliography database is also created and developed throughout the development of WebBiblio, and it is used in the verification, validation and testing of software system. The development of the database is explained in detail in Section 4.3.1 on page 72. Currently, the database contains 534 bibliographic references and more than 500 of these are with their abstracts and keywords. WebBiblio has been publishing the Simulation VV&T Bibliography since July 1, 1996.

Additional to Simulation VV&T Bibliography, as part of DMSO research project, also Expert System/Knowledge-Based System VV&T Bibliography is created and developed, which WebBiblio has been also publishing since January, 1997.

CHAPTER 7 : SUMMARY AND CONCLUSIONS

This thesis research has produced WebBiblio, a software system publishing bibliographies globally on the WWW. Considering the features provided, it is a unique WWW software in bibliography publishing. WebBiblio is not a prototype, but a working, complete tool. It is currently publishing three VV&T Bibliographies.

This final chapter gives the summary of the thesis by describing the contents of each chapter, makes the conclusions of the research, and finally provides recommendations for future research.

7.1 Summary of the Thesis

This thesis describes the software engineering of a software system, WebBiblio, that can be used for publishing bibliographies globally on the WWW with a rich set of features.

Chapter 1 introduces the conducted thesis research. It states the problem that the study focuses on, and defines the objectives of the research in order to provide a comprehensive solution for the stated problem. Finally the chapter provides an overview of the thesis.

Chapter 2 presents the literature review. It compares the features and capabilities of WebBiblio with other software products on the WWW publishing bibliographies. A table summarizing the comparison of the software products discussed in the chapter with WebBiblio is given at the end of the chapter.

Chapter 3 first lists the requirements specification of the software system with respect to the objectives of the study. Then it introduces the hardware and software environments used throughout the development of the software product. Finally it describes the design specification and implementation of WebBiblio.

Chapter 4 explains every component and feature of WebBiblio in detail with screen-shots and examples.

Chapter 5 provides guidance to WebBiblio system administrators by explaining the regular administrative tasks in a specific, step by step format.

Chapter 6 makes the subjective evaluation of the study with respect to study objectives.

Chapter 7 first summarizes the contents of the thesis, then makes the conclusions and gives recommendations for future research.

7.2 Conclusions

This thesis deals with the development of a software system that is publishing bibliographies globally on the WWW. The research aims the development of a complete software product rather than a prototype.

WebBiblio is developed mainly based on waterfall software life-cycle model [Schach 1996]. As initial step, the problem that is addressed in the study is clearly defined. Then the other available software products providing solutions for the same problem are carefully searched, reviewed and evaluated. With the guidance of these evaluations, the objectives of WebBiblio are defined to develop a product with a unique set of features and

capabilities, and consequently to provide an comprehensive solution for the addressed problem. Based on these objectives, the requirements of WebBiblio are specified With respect to these specifications, the software and hardware components of the system and their interactions are determined, high level design decision are made. Following that, the high level design is expanded into the low level design. The syntax of *Query Form* and its every field, the structure of the search engine, the structure and contents of data files, user interface of each component, and contents and structure of *Help* are defined.

Parallel to the implementation of the software system, the bibliographies to be published by WebBiblio as part of DMSO project are created and their databases are continuously improved with additions. During the implementation, each unit of the system is tested and validated immediately after its implementation by using Simulation VV&T Bibliography database, which is also being improved at the same time. When the implementation of the system components are complete, and the integration and system tests are performed, the bibliography database have also become reach enough to perform complete functionality testing of WebBiblio on.

By the end of August 1996, the implementation, verification, validation and testing of WebBiblio with all its components were complete. Since September 1, 1996, it has been publishing officially the Simulation and Software VV&T bibliographies both with more than 500 references. It has been also publishing Expert System/Knowledge-Based System VV&T Bibliography since January, 1997.

Currently, WebBiblio is a working tool that is globally accessible on the WWW. It has been used by a wide variety of users from different organizations and countries as well

as by the CS6704[1] students for their class projects involving literature search over the bibliographies published by WebBiblio (Appendix A on page 113 gives more detailed information about the access statistics). It is currently publishing three VV&T bibliographies as stated above that are determined based on the DMSO research project which is also addressed in this thesis. However, WebBiblio is a general software product that can publish any type of scientific bibliography independent of the DBMS that is used for the maintenance of the bibliography databases as long as the data files can be created properly in required format from these databases. Hence, any searchable scientific bibliography can be published by WebBiblio within two hours after the bibliography has received in the correct form of data files.

To conclude, WebBiblio, the developed software system in the frame of this research, is a software product serving its users by publishing bibliographies on the WWW in a execution efficient, easily and globally accessible way. With its well human engineered interface, advanced query form providing a wide variety of query composition capabilities over any field of reference structure as well as abstracts and keywords, structured help utility explaining every component of the system in detail, and other complementary pages, today WebBiblio makes a contribution to computer science at this cutting-edge of the technology by providing a comprehensive solution to the problem of global publication of bibliographies.

[1] CS6704: Software Verification, Validation and Testing. Department of Computer Science, Virginia Tech, Fall 1996. Taught by Dr.Osman Balci.

7.3 Recommendations for Future Research

Recommendations for future research can be grouped into two categories: increasing (1) the capabilities of *Query Form*, and (2) the usability of the software system.

7.3.1 Increasing the Capabilities of Query Form

The capabilities of *Query Form* can be expanded by adding more functionality. The PERL regular expressions [Quigley 1995] can be allowed in the query composition syntax of the form fields that support string search over the related reference fields. These expressions can be syntactically distinguished in the query by requiring the user to place them, for instance, in square brackets. Since PERL is the implementation language of the search engine running at the background, after the regular expressions are properly parsed, extension of the searching process with their inclusion is relatively simple. It only requires passing the search string with its attached regular expressions as input parameter to the search function; no further processing on regular expressions is necessary.

The searching capabilities can be further improved by allowing misspellings in the query composition. Hence, the searched key and the matching information may vary in or less than a certain number of characters, where this number is determined by the user as part of the query. Because of the structure of data files used in the search operation, this improvement requires complicated computations and drastic increases in response time should be expected. Unless the structures of data files are significantly changed, or some other special data files are created for the purpose of this extension, the execution time of

the search engine is expected to increase exponentially with increasing allowed number of misspelled characters.

Another new feature that can be added to the capabilities of *Query Form* is providing the option of performing the search operation over multiple bibliographies rather than only over the bibliography whose *Query Form* is currently being used. Hence, *Query Form* of each bibliography lists the names of other bibliographies published at the same WWW site with attached checkboxes next to their names. By checking these boxes, the user can expand the range of search operation based on his/her query over multiple bibliographies.

Currently, when the user reloads *Query Form* to modify his/her query, the last composed search query may or may not still exist on the new form depending on the capabilities of the web browser used by the user. By improving the search engine, the existence of the last composed query on the reloaded form for easier improvement of queries can be guaranteed.

7.3.2 Increasing the Usability of Software System

By adding new features to WebBiblio when displaying search results, the usability of the software can be increased. A new feature can be to allow manually selection of some references from the result display. With this feature, a user can narrow the search results even further so that the final reference list contains only those references the user is interested in. This allows the user to concentrate on certain references more easily. The feature may also simplify the resulting documentation. For instance if the page is printed it

only contains the list of specifically selected references. To implement this additional feature, a checkbox can be attached next to each reference information in the query results listing. The user checks these boxes to make his/her selections from the list. When the web page is reloaded, it only displays the selected references from the initial list.

This feature can be even further improved by providing the capability to combine the list of manually selected references from different search query results in one listing. That can be done by providing a general bucket for each user. The user can put any reference any time into this bucket by selecting it from the display of search results, and the content of the bucket does not change when the user runs new search queries. Finally, the user can view the references in the bucket as a list any time by clicking on a special button. The implementation of this feature introduces two problems to be solved. First the users of WebBiblio must be distinctly identified. This is necessary to be able to assign a bucket file to every user. Second, the bucket files must be deleted when they are obsolete. To solve this problem, it must be detected by the system when the user quits using WebBiblio, so the user's bucket file can be safely deleted. An alternative solution for second problem can be assignment of a certain life span to the bucket files, for instance one day. Hence, they can be deleted one day after they are created.

Subdivision of the query results display into multiple pages, whose size is defined by the user in terms of the number of the references on each page, and support of connections to mail applications to send e-mails containing the search results can be other improvements to WebBiblio increasing its usability.

Currently, when a bibliography is selected from the web page that is listing the published bibliographies, WebBiblio downloads *Query Form* of the selected bibliography. Instead of *Query Form*, a special information page that is created separately for each bibliography to provide introductory information about the bibliography and the usage of WebBiblio can be initially downloaded to increase the system's usability further.

Finally, the usability of WebBiblio can be increased even further by providing links between the forms and *Help* utility for a much faster on-line assistance. For instance, each field name on *Query Form* and *Contribution Form* can be a link to the section of *Help* that contains the help information about this form field.

REFERENCES

Achilles, A.-C. (1996a), “The Collection of Computer Science Bibliographies”, http://liinwww.ira.uka.de/bibliography/index.html.

Achilles, A.-C. (1996b), “Help on the Computer Science Bibliography Collection Advanced Search Facility”, http://liinwww.ira.uka.de/bibliography/advanced.search.help.html.

Achilles, A.-C. (1996c), “Computer Science Bibliography Collection Advanced Search”, http://liinwww.ira.uka.de/bibliography/waisbib.html.

Achilles, A.-C. (1996d), "Bibliographies on Database Research", http://liinwww.ira.uka.de/bibliography/Database/index.html.

Caere Corporation (1996),*OmniPage Pro User’s Manual for Macintosh*, Caere Corporation, Los Gatos, CA.

Claris Corporation (1992),*FileMaker Pro User’s Guide for Macintosh*, Claris Corporation, Santa Clara, CA.

Constantine, L. L. and E. Yourdon (1979),*Structured Design*, Prentice-Hall, Englewood Cliffs, NJ.

Graham, I. S. (1996),*The HTML Sourcebook: A Complete Guide to HTML 3.0*, John Wiley & Sons, New York, NY, 2nd edition, 688p.

Jones, D. M. (1996), "Searching the HBP Bibliographies", http://theory.lcs.mit.edu/~dmjones/hbp/bibsearch.html.

Kamath, M. U. (1996), "Quick-Search Database Systems Research Bibliography", http://www-ccs.cs.umass.edu/db/bib-search-old.html.

Manber, U., S. Wu and B. Goppal (1996), “Glimpse: A Tool to Search Entire File Systems”, http://glimpse.cs.arizona.edu/index.html.

Nielsen, J. (1990), “Traditional Dialogue Design Applied to Modern User Interfaces”, *Communications of the ACM 33*, 10, 109-118.

OCLC (1996), “Using FirstSearch”, http://www.oclc.org/oclc/man/fs/9289usfs/9289.htm.

Pfeifer, U. (1996a), “freeWAIS-sf”, http://ls6.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf/.

Pfeifer, U., N. Fuhr and T. Huynh (1996b), “Searching Structured Documents with the Enhanced Retrieval Functionality of freeWAIS-sf and Sfgate”, http://www.igd.fhg.de/www/www95/papers/47/fwsf/fwsf.html.

Quigley, E. (1995), *PERL by Example*, Prentice-Hall, Englewood Cliffs, NJ, 358p.

Rijsbergen, J. C. van (U1), "Chapter Four: File Structures", http://www.dcs.glasgow.ac.uk/Keith/Chapter.4/Ch.4.html.

Schach, S.R. (1996), *Classical and Object-Oriented Software Engineering*, Irwin, Chicago, IL, 3rd edition, pp. 53-58.

Sommerville, I. (1996a), *Software Engineering* Addison-Wesley, Reading, MA, 5th edition, pp. 341-343.

Sommerville, I. (1996b), *Software Engineering* Addison-Wesley, Reading, MA, 5th edition, pp. 629-632.

Winkler, D., S. Kamins (1990), *HyperTalk 2.0: The Book*, Bantam Books, New York, NY, 958p.

Yourdon, E. (1996), “Java, the Web, and Software Development”, *Computer 29*, 8, 31-39.

APPENDIX : ACCESS STATISTICS

Some information about the access of WebBiblio is presented in this Appendix. The information is based on the usage of *Query Form* and the requests for the abstracts and/or keywords rather than visitor number of the WebBiblio web site. Hence, the statistics used in the preparation of the charts in this section consists of the execution counts of the programs *bibSearch.pl* and *abskw.pl*, which are the search engine and abstract-keywords engine of WebBiblio, respectively. Each execution of *bibSearch.pl* means submission of a search query through *Query Form*, and each execution of *abskw.pl* means an abstract and/or keywords request through the web page displaying query results. Section 4.1.3 on page 51 and Section 4.1.4 on56 give detailed information about these engines.

When statistical data is collected, only the real users of WebBiblio are considered. In other words, the accesses to the programs by the WebBiblio's developers are ignored. This is managed by not including the accesses from the domains that are used by the development team in the statistical information.

The collection of data was started on July 1, 1996. At this date, only Simulation VV&T Bibliography was available. Software VV&T Bibliography became available on September 1, 1996 and since then access to this bibliography was also recorded. The charts in this appendix contain the access information to these bibliographies starting from the dates when they became globally available to November 30, 1996 (Expert System/Knowledge-Based System VV&T Bibliography became available in January 1997 and it is not used for statistical data collection.) Since each published bibliography has its

own engines, the execution of four programs were recorded. During this period, these programs were accessed (executed) 636 times in total. Figure A.1 displays the distribution of these 636 accesses among the programs. In the figure, the bibliographies that the programs belong to are indicated in the parentheses.

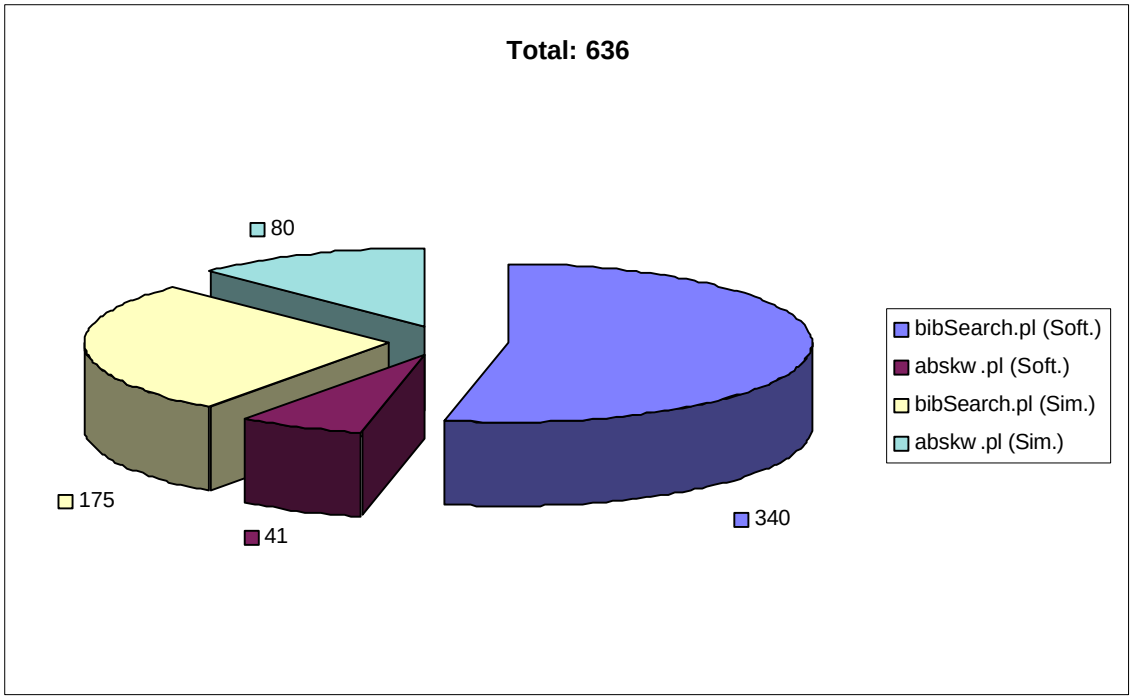

Figure A1 Access Distribution among the four Programs

Figure A.2 displays the access distribution with respect to the months. July and August contain only the access rate of Simulation VV&T Bibliography's programs and the other columns the access rate of all four programs. The relatively high access rate in October is because of the heavy usage of WebBiblio by CS6704[1] students for their class projects involving literature search over the bibliographies published by WebBiblio.

[1] CS6704: Software Verification, Validation and Testing. Department of Computer Science, Virginia Tech, Fall 1996. Taught by Dr.Osman Balci.

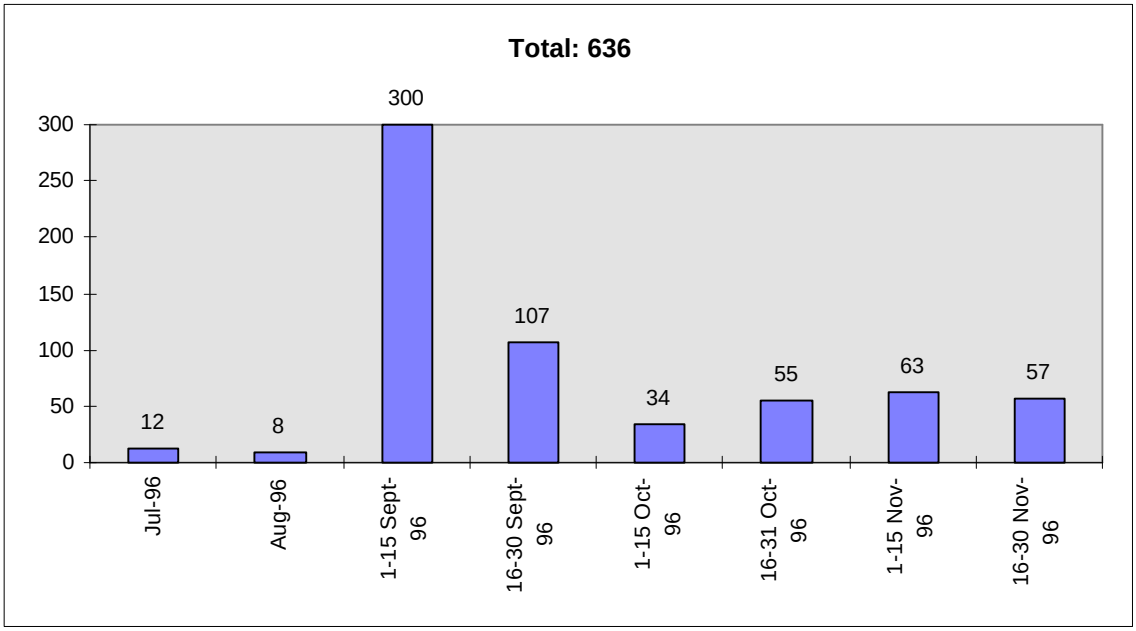

Figure A2 Access Distribution with respect to the Months

Figure A.3 and Figure A.4 display the access distribution with respect to the months at the program level.

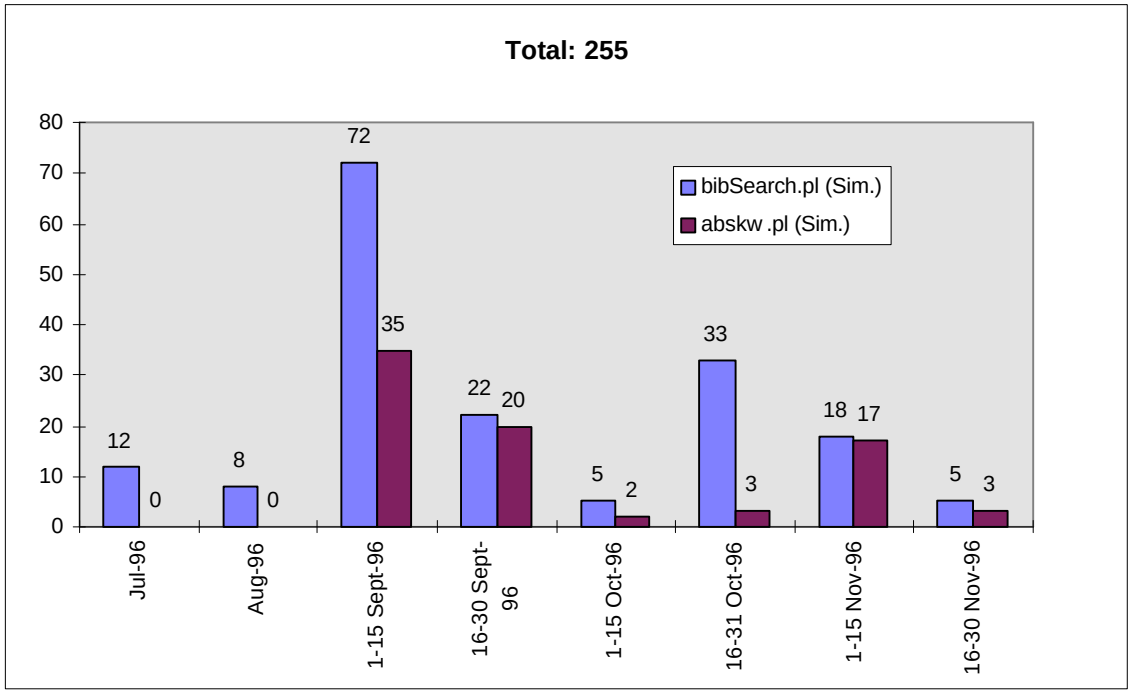

Figure A3 Access Distribution of Simulation VV&T Bibliography

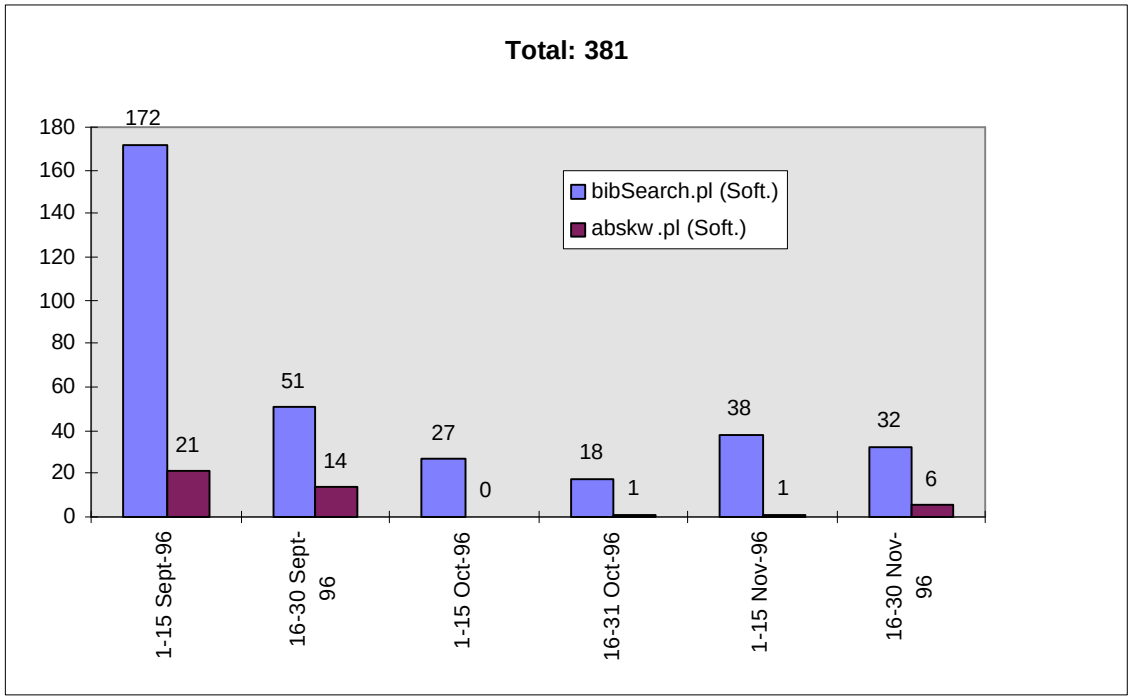

Figure A4 Access Distribution of Software VV&T Bibliography

The chart in Figure A.5 focuses on the variety of WebBiblio users. It summarizes the access distribution with respect the user's organization types, which are determined from the Internet domain name from where the access is made.

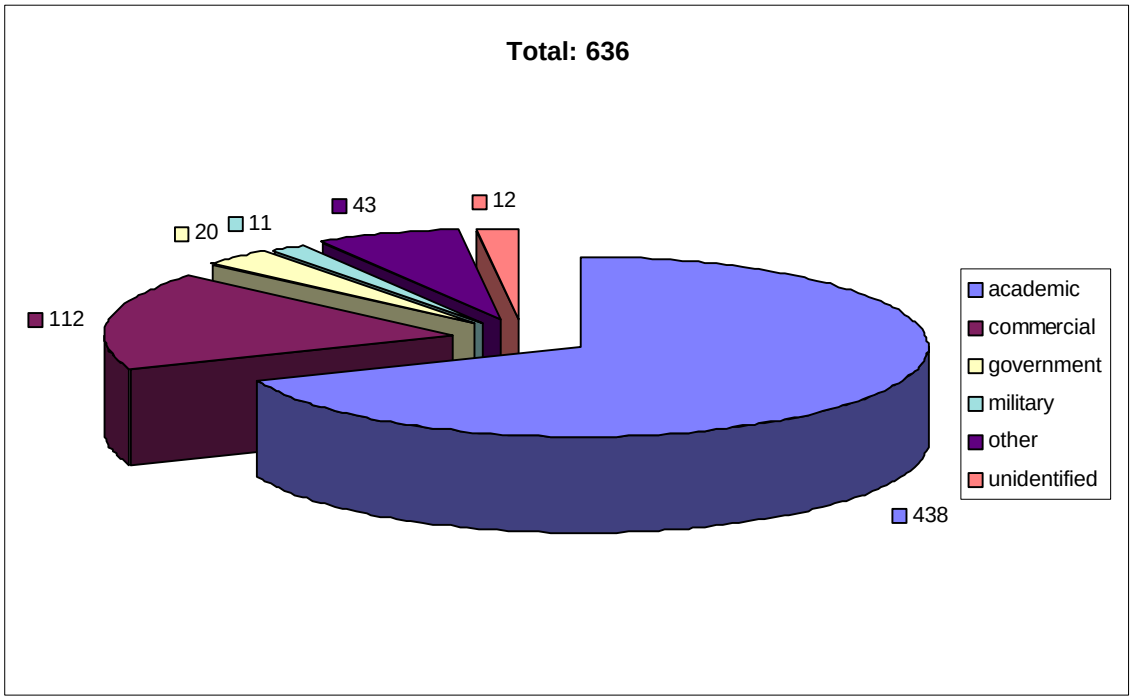

Figure A5 Access Distribution with respect to Internet Domain

Similarly, Table A.1 summarizes the variety of countries from which the bibliographies were accessed. Again, the information is determined based on the domain name from where the access request originates.

Table A.1 Access Distribution with respect to the Countries

	bibSearch.pl (Soft.)	abskw.pl (Soft.)	bibSearch.pl (Sim.)	abskw.pl (Sim.)	Total
USA	322	41	131	73	567
Australia	4	-	16	-	20
France	2	-	8	-	10
Germany	3	-	-	6	9
Finland	6	-	-	-	6
Nicaragua	2	-	1	-	3
Brazil	-	-	3	-	3
Uni. Kingdom	-	-	3	-	3
Belgium	-	-	2	-	2
Sweden	1	-	-	-	1
unidentified	-	-	11	1	12

VITA

Name of Author Ibrahim Utku Moral

Place of Birth Ankara, Turkey

Date of Birth August 29, 1971

Education

Master of Science in Computer Science, 1997, Virginia Tech, Blacksburg, Virginia, USA

Bachelor of Science in Computer Engineering and Information Science, 1994, Bilkent University, Ankara, Turkey.

Awards and Honors

Graduate Assistant, August 94 - December 96, Department of Computer Science, Virginia Tech.

Scholarship Award, September 89 - June 94, Department of Computer Engineering and Information Science, Bilkent University.

Scholarship Award, September 92 - June 94, Turkish National Scientific and Technological Research Council.

1st Prize, May 89, Turkish National Scientific and Technological Research Council Regional High School Chemistry Contest.