INFORMATION TO USERS

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

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

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

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

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

WITH jMap® TECHNOLOGY

Puping Peng

A MAJOR REPORT

IN

THE DEPARTMENT

of

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

APRIL 2002

© PUPING PENG, 2002
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

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

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

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

0-612-68477-6
Abstract

Web Mining With jMap Technology

Puping Peng

This report presents a software development process of using Breadth-First-Search algorithm to discover pages in a web site, retrieve and analyze information from such sites. The information is transformed to \textit{joined maps} for content and structure mining. The software “Website jMap Builder” is the first step towards web mining using jMap technology. It was developed using PERL and Visual Basic. Functional and design specifications of the system are given in Chapter 4 and a user manual is provided in Chapter 5.
Acknowledgements

I would like to express my special thanks to Dr. W.M. Jaworski, for his encouragement, technical advice, constant help and critical remarks throughout this work. The author was deeply benefited from his knowledge about information systems.

I also like to express my thanks to the following individuals and companies:

- Professor Dr. Rajagopalan Jayakumar for acceptance as my co-supervisor and his useful comments for this report.
- Ms. Halina Monkiewicz, the Graduate Program Secretary, for her kindly support.
- My classmate and former colleague, Minghui Han, for his valuable input and help.
- Motorola Canada Software Center and Siemens Canada Ltd. for their generous financial supports during my study.

Finally, I am deeply indebted to my family for their continuous supports and encouragement.
Table of Contents

LIST OF FIGURES .......................................................................................................................... V

LIST OF TABLES ............................................................................................................................ VI

1. INTRODUCTION ...................................................................................................................... 1
   1.1 BACKGROUND .................................................................................................................... 1
   1.2 OBJECTIVE OF STUDY ..................................................................................................... 5
   1.3 PROJECT SCOPE ............................................................................................................... 6

2. WEB SITE ORGANIZATION AND REPRESENTATION .......................................................... 7
   2.1 GRAPH NOTATION .......................................................................................................... 7
      2.1.1 Representing Web as Graph ....................................................................................... 7
      2.1.2 Web Organization Models ....................................................................................... 8
   2.2 UML NOTATION ............................................................................................................. 13
      2.2.1 UML Web Extension ............................................................................................... 13
      2.2.2 Web Design Modelling with UML ......................................................................... 17
   2.3 CONTEXT MAP NOTATION ............................................................................................... 18
      2.3.1 Context Maps .......................................................................................................... 18
      2.3.2 Joined Maps (JMaps) ............................................................................................... 19
      2.3.3 jMap Syntax and Process ....................................................................................... 20
      2.3.4 jMap Notation ........................................................................................................ 22
      2.3.5 Representing Models in jMap Notation ................................................................... 25
   2.4 THE ANALYSIS OF JMAP WEB MODEL ....................................................................... 29

3. WEB MINING WITH JMAPS ................................................................................................. 30
   3.1 WEBSITE STRUCTURE GRAPH MODEL TO JMAP MODEL MAPPING ......................... 30
   3.2 WEBSITE STRUCTURE RECOVERING USING BFS ....................................................... 30
      3.2.1 Breadth-First-Search Algorithm (BFS) .................................................................. 30
      3.2.2 Generic Website jMap Schema .............................................................................. 33
      3.2.3 Mapping Website Attributes and BFS tree attributes to jMap Concepts ............... 34
      3.2.4 A Website Model Recovered with JMaps ............................................................... 35
   3.3 DEVELOPMENT TOOLS ................................................................................................. 36
   3.4 WEBSITE JMAP BUILDER DESIGN CONSTRAINTS ...................................................... 37

4. SOFTWARE SPECIFICATIONS ................................................................................................. 38
   4.1 ARCHITECTURE ............................................................................................................ 38
   4.2 FUNCTIONALITIES ......................................................................................................... 39
   4.3 SOFTWARE DESIGN ....................................................................................................... 40
      4.3.1 Architecture Design ............................................................................................... 40
      4.3.2 Detailed Design ...................................................................................................... 41

5. SOFTWARE USER MANUAL .................................................................................................. 48
   5.1 SYSTEM REQUIREMENTS .............................................................................................. 48
   5.2 SOFTWARE INSTALLATION ............................................................................................ 49
      5.2.1 PERL 5.005 ............................................................................................................ 49
      5.2.2 PERL Libraries ....................................................................................................... 49
      5.2.3 Microsoft Excel ....................................................................................................... 49
      5.2.4 Website JMap Builder Software ........................................................................... 50
   5.3 LOCAL TERMINAL APPLICATION ................................................................................. 51
      5.3.1 Run Local Terminal Application on a PC ............................................................. 51
      5.3.2 Run Local Terminal Application on UNIX/Linux .................................................. 54
List of Figures

Figure 1 Sequential Organization of Web Pages ......................................................... 9
Figure 2 Grid Organization of Web Pages ...................................................................... 11
Figure 3 Hierarchy Organization of Web Pages ............................................................. 11
Figure 4 Web-like Organization of Web Pages ............................................................. 12
Figure 5 Comparison Chart of Web Organization Complexity ....................................... 13
Figure 6 Example of UML Web Architecture Design .................................................... 18
Figure 7 Diagrams Defined by Map Patterns .............................................................. 21
Figure 8 Schema view of Map ....................................................................................... 22
Figure 9 Customer and Orders Model Represented by jMap .......................................... 27
Figure 10 A Web Design jMap Model Converted from Figure 6 UML model ............... 29
Figure 11 Generic jMaps Template for Website Model .................................................. 34
Figure 12 A Tailored Version of Web Site Model for www.cs.concordia.ca ..................... 36
Figure 13 Web jMap Builder Architecture ...................................................................... 38
Figure 14 Software Architecture Design Diagram ....................................................... 41
Figure 15 Website jMap Builder Library Classes ......................................................... 42
Figure 16 MS Excel Macros Interface Worksheet .......................................................... 52
Figure 17 "Website jMap Builder" GUI ........................................................................ 53
Figure 18 Open Web Site jMap Building Report ........................................................... 59
Figure 19 Enter Session Number to Open jMap ............................................................ 60
Figure 20 Formatting jMap Results .............................................................................. 60
**List of Tables**

Table 1  Properties of A Web Page................................................................. 7  
Table 2  Extended Stereotyped Classes Introduced to UML Web Design .......... 14  
Table 3  Stereotyped Associations Introduced to UML Web Design ................ 15  
Table 4  Stereotyped Attributes Introduced to UML for Web Design .............. 16  
Table 5  A jMap Schema for Representing Web Designs .................................. 28  
Table 6  Website Attribute Mapping To jMap Concepts .................................. 35  
Table 7  Web Site jMap Builder Directory Structure and Files .......................... 50
1. Introduction

1.1 Background

The widespread development of internet has made the web sites the largest mass media next to the television. This growing trend will make the web the number one media in the near future. With the explosive growth of information sources available on the World Wide Web, it has become increasingly necessary for users to utilize automated tools in finding the desired information resources, and to track and analyze their usage patterns. These factors give rise to the necessity of creating server-side and client-side intelligent systems that can effectively mine for knowledge[1-3].

Web Mining is the extraction of interesting and potentially useful patterns and implicit information from artifacts or activity related to the World-wide Web. There are roughly three knowledge discovery domains that pertain to web mining: Web Content Mining, Web Structure Mining, and Web Usage Mining. Web content mining is the process of extracting knowledge from the content of documents or their descriptions. Web document text mining, resource discovery based on concepts indexing or agent-based technology may also fall in this category. Web structure mining is the process of inferring knowledge from the World-Wide Web organization and links between references and referents in the Web. Finally, web usage mining, also known as Web Log Mining, is the process of extracting interesting patterns from web access logs.

Web Content Mining
Web content mining is an automatic process that goes beyond keyword extraction. Since the content of a text document presents no machine-readable semantic, some approaches have suggested to restructure the document content in a representation that could be exploited by machines. The usual approach to exploit known structure in documents is to use wrappers to map documents to some data model. Techniques using lexicons for content interpretation are yet to come. There are two groups of web content mining strategies: Those that directly mine the content of documents and those that improve on the content search of other tools like search engines.

**Web Structure Mining**

World-Wide Web can reveal more information than just the information contained in documents. For example, links pointing to a document indicate the popularity of the document, while links coming out of a document indicate the richness or perhaps the variety of topics covered in the document. This can be compared to bibliographical citations. When a paper is cited often, it ought to be important. Some software already uses these methods to take advantage of this information conveyed by the links to find pertinent web pages. By means of counters, higher levels cumulate the number of artifacts subsumed by the concepts they hold. Counters of hyperlinks, in and out documents, retrace the structure of the web artifacts summarized.

**Web Usage Mining**

Web servers record and accumulate data about user interactions whenever requests for resources are received. Analyzing the web access logs of different web sites can help understand the user behaviour and the web structure, thereby improving the design of this
colossal collection of resources. There are two main tendencies in Web Usage Mining driven by the applications of the discoveries: General Access Pattern Tracking and Customized Usage Tracking. The general access pattern tracking analyzes the web logs to understand access patterns and trends. These analyses can shed light on better structure and grouping of resource providers. Many web analysis tools have been designed to make use of web logs. Applying data mining techniques on access logs unveils interesting access patterns that can be used to restructure sites in a more efficient grouping, pinpoint effective advertising locations, and target specific users for specific selling ads. Customized usage tracking analyzes individual trends. Its purpose is to customize web sites to users. The information displayed, the depth of the site structure and the format of the resources can all be dynamically customized for each user over time, based on their access patterns. While it is encouraging and exciting to see the various potential applications of web log file analysis, it is important to know that the success of such applications depends on what and how much valid and reliable knowledge one can discover from the large raw log data. Although current web servers store limited information about the accesses, but the trend to have more logs is growing. Some scripts custom-tailored for some sites may store additional information.

Web mining is interdisciplinary in nature, cutting across such fields as information retrieval, information representation, natural language (information) processing, machine learning, database, and visualization. The web mining technology will yield benefits to at least the following fields:

- Web classification and context based search engine[4, 15].
• Web site construction and site performance tuning.

• Site effective browsing and navigation.

• User analysis for better marketing strategies

There are three challenging steps for web mining technologies. The first is to harvest information from immense web-like resources. The second step is to reorganize this tremendous amount of information in proper notations for processing. The last step is to analyze this information to produce useful results.

Currently, the majority of internet resources are open source web sites, which are made of more than 80% of HTML pages. A small portion of information is embedded in special scripts and multimedia files such as flash, sound and metadata, etc. The importance of analysis of HTML pages is obvious. The HTTP protocol is an open protocol; this make possible to make use of it to explore the web sites. There is tremendous amount of library resources have been developed to retrieve and analyze information from the Internet. One of the most abundant library resources available is for the open source language PERL. These open source libraries made much easier for developers to write tools automatically retrieving the web pages and analyzing them.

There are many representations and methodologies for information systems such as graphical notations for the information system representation and UML notation for Object oriented systems design [5]. However it is a challenge to develop a notation with good processability and simple and easy to use.
The *joined maps* viewed as *context maps* in this report is one of the ways to satisfy the above requirements. The *joined maps*, or *jMaps*, is a notation and a method for representing systems architecture, structures, processes and reusable templates. This technology was first introduced by W.M. Jaworski [6-12]. The technology was initially developed as a means of recovering and refining knowledge from legacy system. With *jMaps* or *Context Maps* technology, which uses the popular concept of a spreadsheet to represent the recovered information, thus make its represented contents easily processable. The *jMaps* notation allows efficient and complete representations of a web site contents and modeling of generic schemata for web-based information systems. It is one of the ideal technologies will render benefits for web mining process. This research is a continuing part of the researches to automate the web mining process by using jMap technology under Dr. Jaworski’s supervision.

1.2 Objective of Study

The main objective of this project is to automate information recovery from a website and analyze each page using the predefined jMap template and convert the discovered information to jMap notation for web content and structure mining. The subsequent mining of recovered knowledge in jMaps is beyond the scope of current research.

The main thrust of this software was based on BFS algorithm and jMap notation technology and implemented in PERL/VBA. where the PERL program accesses web pages and anaylze contents and the VBA program is used to format the analysis results to Microsoft Excel spreadsheets. The result jMap was formatted in Microsoft Excel by
Excel Macros and ready to be processed by using jMap Context analysis programs. The main program and GUI run on UNIX, Linux and Windows systems. However, it formats the results to jMaps only on computers equipped with MS Excel.

1.3 Project Scope

The "Web site jMap builder" project started in the fall of 2000, under the supervision of Prof. Jaworski. The initial scope of this project is the following:

1) To understand jMap web site model and study the possibility of automating jMapping a web sites by a software program.

2) To identify suitable template for web site content and structure recovery.

3) To find suitable algorithms to map a web site and analyze pages to generate jMaps.

4) To transform recovered Web site Information to jMap and format the jMap results to Microsoft Excel for easy processing.

5) To provide basic GUI for user to specify websites, maximum depth, and maximum number of retrieved URLs.

6) To enable crossing site mapping.

7) To provide basic documentation for requirements, design and implementation, user manual.

8) To summarize project and make suggestions for future research.
2. Web Site Organization and Representation

2.1 Graph Notation

2.1.1 Representing Web as Graph

The graph notation is one of the fundamental mathematical notation and the most intuitive notation to represent a web site content and its organizing structure. We can represent a page using a vertex, and their relationship using directed edges.

One of the advantages of graph model for web site contents is its simplicity. Each HTML page can be viewed as a vertex in the graph, and all the contents can be viewed as the properties of this vertex. The hyperlinks serve as the relationship between different web pages. The script links can be viewed as the conditional relationship between pages. Table 1 Properties of A Web Page lists some of the properties of web pages and their notation in a graph.

Table 1 Properties of A Web Page

<table>
<thead>
<tr>
<th>Property</th>
<th>Name</th>
<th>Graph Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistical</td>
<td>Number of bookmark.</td>
<td>vertex property</td>
</tr>
<tr>
<td>Size of the page</td>
<td></td>
<td>vertex property</td>
</tr>
<tr>
<td>Number of hyperlinks out to other pages from this site</td>
<td></td>
<td>vertex property</td>
</tr>
<tr>
<td>Number of hyperlinks linked to this page from this site</td>
<td></td>
<td>vertex property</td>
</tr>
<tr>
<td>Dynamic</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>Retrieval Time</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td>Server Return code for this page</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td>Meta Text: Key words</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td>HTML texts</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td>Images</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td>hyperlink token text/image</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td>Forms, Frames and Other Advanced Elements</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td>Multimedia Contents such as sound, flash media</td>
<td>vertex property</td>
<td></td>
</tr>
<tr>
<td><strong>Relationship</strong></td>
<td>self-directed edge</td>
<td></td>
</tr>
<tr>
<td>bookmark</td>
<td>self-directed edge</td>
<td></td>
</tr>
<tr>
<td>hyperlinks/image maps</td>
<td>directed edges</td>
<td></td>
</tr>
<tr>
<td>Scripts: ASP, CGI, JAVASCript, Applet, VBA, and other technologies</td>
<td>directed edges with conditions</td>
<td></td>
</tr>
</tbody>
</table>

### 2.1.2 Web Organization Models

As far as World Wide Web is concerned, you can hardly escape references to hypertext and hypermedia. In the epoch of computer press, there is a lot of fuzzy thinking about how web-based information can somehow "link everything to everything." The implication is that with the web you can probably dispense with one of the most challenging aspects of presenting information -- how to put it into logical order and create
an interesting and understandable resource for the user. Nothing could be further from the truth. If a web site has no comprehensive narrative or clear sense of organization, the readers will probably know it soon enough, and most of them will leave in pursuit of better organized material. Because of this reason, the concept of web site organization is rapidly evolving, and novel design concepts are sprouting in every corner of the cyberspace. However, from a graphical point of view, the center of the web design models falls into four categories: sequential model, grid model, hierarchy model and web model.

2.1.2.1 Sequence Model

The simplest way to organize information is in a sequence, where you present a linear narrative. Information that naturally flows as a narrative, time line, or in logical order is ideal for sequential treatment. Sequential ordering may be chronological, a logical series of topics progressing from the general to the specific, or even alphabetically sequenced, as in indexes, encyclopedias, and glossaries. However, simple sequential organization usually only works for smaller sites (or structured lists like indexes), as long narrative sequences often become more complex, and thus require more structure to remain understandable.

![Figure 1 Sequential Organization of Web Pages](image-url)
More complex Web sites may still be organized as a sequence, but each page in the main sequence may have one or more pages of digressions, parenthetic information, or links to information in other Web sites.

2.1.2.2 Grid Model

Many procedural manuals, lists of university courses, or medical case descriptions are best organized as a grid. Grids are a good way to correlate variables, such as a time line versus historical information in a number of standard categories such as "events," "technology," "culture," etc. To be successful, the individual units in a grid must share a highly uniform structure of topics and subtopics. The topics often have no particular hierarchy of importance. For example, "tuberculosis" is not more or less important a diagnosis than "pneumonia", but ideally both case descriptions would share a uniform structure of subtopics. Thus, the user could follow the grid "down," reading about tuberculosis, or cut "across" the grid perhaps by comparing the "diagnostic imaging" sections of both pneumonia and tuberculosis. Unfortunately, grids can be difficult to understand unless the user recognizes the interrelationships between categories of information, and so are probably best for experienced audiences who already have a basic understanding of the topic and its organization. Graphic overview maps normally accompany in grid-like Web sites.
2.1.2.3 Hierarchy Model

Information hierarchies are one of the best ways to organize complex bodies of information. Hierarchical organization schemes are particularly well-suited to Web sites, because Web sites should always be organized as off-shoots of a single home page. Most users are familiar with hierarchical diagrams, and find the metaphor easy to understand as a navigational aid. A hierarchical organization also imposes a useful discipline on web site content design, as hierarchies only work well with highly organized material. Since hierarchical diagrams are so familiar in corporate and institutional life, users find it easy to build mental models of the site:

Figure 3 Hierarchy Organization of Web Pages
2.1.2.4 Web Model

Web-like organizational structures pose few restrictions on the pattern of information use. The goal is often to mimic associative thought and free flow of ideas, where users follow their interests in a heuristic, idiosyncratic pattern unique to each person who visits the site. This organizational pattern develops in Web sites with very dense links both to other information within the site, and information on other World Wide Web sites. The goal is to fully exploit the Web’s power of linkage and association, but web-like organization structures can just as easily propagate confusion and fuzzy thinking about the interrelationships of web site information chunks. Ironically, organizational webs are often the most impractical structure for Web sites, because they are so hard for the user to understand and predict. Webs work best for small sites dominated by the number of links, aimed at highly educated or experienced users looking for enriching user’s knowledge rather than for a basic understanding of new topic.

Figure 4 Web-like Organization of Web Pages
2.1.2.5 Summary

Most complex web sites share aspects of all four types of information structures except in sites that rigorously enforce a single model. The ultimate goal of a web site design is to give the users a clear, consistent and navigatable structure and content. The chart below summarizes the four basic organization patterns against the "linearity" of your narrative, and the complexity of your content.

![Comparison Chart of Web Organization Complexity](image)

Figure 5 Comparison Chart of Web Organization Complexity

2.2 UML Notation

2.2.1 UML Web Extension

UML is a visual notation designed for object-oriented design concept[5] and a notation with system design in mind. It is widely used for communicating system design ideas and implementation details. In order to accommodate Web design concept, OMG and
Rational has introduced a new set of extensions to the UML notation[21-23], which defines a set of stereotypes, tagged values and constraints that are suitable for modeling web applications. The stereotypes and constraints are applied to certain components that are particular to web applications and allows us to represent them in the same model, and on the same diagrams that describe the rest of the system.

2.2.1.1 Stereotypes Classes

UML stereotyped classes are used to represent web components both on server side and client side. Table 2 describes the extensions.

Table 2 Extended Stereotyped Classes Introduced to UML Web Design

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Page</td>
<td>A server page represents a web page that has scripts that are executed by the server</td>
</tr>
<tr>
<td>Client Page</td>
<td>An instance of a client page is an HTML formatted web page. Client pages are rendered by client browsers, and may contain scripts that are interpreted by the browser.</td>
</tr>
<tr>
<td>Form</td>
<td>A class stereotyped as a «form» is a collection of input fields that are part of a client page. A form class maps directly to the HTML form tag. Its attributes represent the HTML form's input fields.</td>
</tr>
<tr>
<td>Frame Set</td>
<td>A frame set is a container of multiple web pages. The rectangular viewing area is divided up into smaller rectangular frames. A frame set stereotyped class maps directly to a frameset web page, and the HTML frame tag.</td>
</tr>
<tr>
<td>Target</td>
<td>A target is a named compartment in a browser window where web pages can be rendered in. Typically a target is one frame in a window defined by a frameset, however a target could be a completely new browser instance or window. «Targeted link» associations specify targets as the place where a new web page is to be rendered.</td>
</tr>
<tr>
<td>JavaScript</td>
<td>On a JavaScript enabled browser it is possible to simulate user defined objects with JavaScript functions. «JavaScript» instances exist only in the context of client pages.</td>
</tr>
</tbody>
</table>
2.2.1.2 Stereotypes Associations

Stereotyped associations were introduced to describe the relationship between web classes. Currently Rational introduced 9 different types of associations to UML notation as in Table 3 Stereotyped Associations Introduced to UML Web Design.

Table 3 Stereotyped Associations Introduced to UML Web Design

<table>
<thead>
<tr>
<th>Association Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>A link is a pointer from a client page to another «Page». In a class diagram a link is an association between a «client page» and either another «client page» or a «server page». A Link association maps directly to the HTML anchor tag. A list of parameter names that should be passed along with the request for the linked page.</td>
</tr>
<tr>
<td>Targeted Link</td>
<td>Similar to a «link» association, a «targeted link» is a link where the associated page is rendered in another target. This association maps directly to the HTML anchor tag, with the target specified by the tag’s target attribute.</td>
</tr>
<tr>
<td>Frame Content</td>
<td>A frame content association is an aggregation association that is expressing a frame’s containment of another page or target. A frame content association can also point to another frameset, indicating nested frames.</td>
</tr>
<tr>
<td>Submit</td>
<td>A «submit» association is always between a «form» and a «server page». Forms submit their field values to the server through «server pages» for processing. The web server processes the «server page», which accepts and uses the information in the submitted form. This relationship indicates which page (or pages) can process the form, and which forms a «server page» has some knowledge about. A list of parameter names that should be passed along with the request for the linked page.</td>
</tr>
<tr>
<td>Builds</td>
<td>The «builds» relationship is a special relationship that bridges the gap between client and server pages. Server pages only exist on the server. They are used to build client pages. The «builds» association identifies which server page is responsible for the creation of a client page. This is a directional relationship, since the client page contains no knowledge of how it came into existence. A server page can build multiple client pages, but a client page can only be built by one server page.</td>
</tr>
<tr>
<td>Redirect</td>
<td>A «redirect» relationship is a unidirectional association with another web page. It can be directed both from and to client and server pages.</td>
</tr>
</tbody>
</table>
Object  | An association between a client page and an object that is embedded in it. The object is typically a Java Applet or ActiveX control. This association maps in part to the HTML <object> tag.
---|---
IIOP | IIOP (Internet Inter-Orb Protocol) is a special type of relationship between objects on the client and objects on the server. It represents a communication mechanism other than HTTP for client server communications. Typically this relationship is between Java Beans on the client and Enterprise Java Beans on the server.
RMI | RMI (Remote Method Invocation) is a mechanism for Java Applets and Beans to send messages to other Java Beans on different machines. Typically this relationship is between Java Beans or Applets on the client and Enterprise Java Beans on the server.

2.2.1.3 Stereotypes Attributes

Four special attributes were introduced to UML notation to describe properties of web pages.

Table 4 Stereotyped Attributes Introduced to UML for Web Design

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Element</td>
<td>An Input Element is an attribute of a «Form» object. It maps directly to the &lt;input&gt; tag in an HTML form. It is used to input of a single word or line of text. The tagged values associated with this stereotyped attribute correspond to the &lt;input&gt; tag's attributes.</td>
</tr>
<tr>
<td>Select Element</td>
<td>An input control used in forms. The select control allows the user to select one or more items from a list. Most browsers render this control as a combo or list box.</td>
</tr>
<tr>
<td>Text Area Element</td>
<td>An input control used in forms, that allows multiple line input.</td>
</tr>
<tr>
<td>Page</td>
<td>A page component is a web page. It can be requested by name by a browser. A Page component may or may not contain client or server scripts. Typically Page components are text files accessible by the web server, but they can also be compiled modules that are loaded and invoked by the web server. Ultimately when accessed by the web server (as either a file or executable) a page produces an HTML formatted document that is send in response to a browser’s request.</td>
</tr>
</tbody>
</table>
2.2.2 Web Design Modelling with UML

UML is widely used for system design. UML web extension allows web design in a way that specifies the underline implementation technology and programming design. It does not point any way of web content structure and logical cohesions. Figure 6 is an example of UML web architecture design that shows an example of using UML for web site design. In this example, UML and UML web extension were used to describe architecture design of a web site.

From this example, we can see UML is useful for system design and implementation, but, it is not convenient and effective for representing web site contents and its structure, and nor does it render any easy processability to the model. It is not a proper notation to represent a web site for knowledge recovery and mining.
Figure 6 Example of UML Web Architecture Design

2.3 Context Map Notation

2.3.1 Context Maps

Context maps were developed by Dr. Jaworski at the University of Concordia[6-13]. This technology has a history of names. During the late 1970s and early 1980s, based on conceptual graphs introduced by J.F.Sowa, it was named as ABL, or Array Based Language [11]. In the late 1980s, it was renamed as ABL/W4. W4 represents as what, when, where and why. In the early 1990s, Prof. Jaworski, by considering existing
notations and methodologies, named this technology as jMap Notation. In the late 1990s until now, jMaps are presented as Context Maps [12].

*Context maps* uses style sheets to control knowledge-based information access and navigation and creates virtual information maps for the information system[9, 12]. An information set is a collection of information classes which contains information instances in a information system[12]. In a sense, *Context maps* describe what an information set is about, by formally declaring topics, and by linking the relevant parts of the information set to the appropriate topics.

Context tuple is a generic association of set members cast in roles. In the extended spreadsheet a column of roles and its related set members define context tuple. From the graphical view, context tuple, in fact, is represented by a compound edge and the connected compound nodes. A directed edge object consists of tail object, middle object and head object. Context can be defined by an aggregation of context tuples. While context tuples represents action-able system behaviors, processes, tasks, procedures or programs. The aggregated context tuples will form a *context map*.

### 2.3.2 Joined Maps (jMaps)

The *joined maps* or *jMaps* is a notation and method for representing systems architecture, structures, processes and reusable templates[12 ]. This technology was first introduced by Dr. W.M. Jaworski [10]. The technology was initially developed as a means of recovering and refining knowledge from legacy systems. By using the popular concept of
spreadsheet structure, it is feasible to describe and process conceptual information. The \textit{jMaps} notation allows efficient recovery and modeling of generic schemata for processes, objects and views of information systems.

\textit{jMaps} represents the knowledge in a spreadsheet format with the relationships represented by vertical \textit{tuples/columns}. Connecting the words jointed and map produced the term "\textit{jMap}". The \textit{jMaps} represent the relationship between different information nodes and provide functionality of arrays, graphs, relational tables, etc. The 'j' stands for \textit{jointed}, because a \textit{jMap} can be a collection of different information connected together in a strong logical way. By that we mean that you can manipulate the logical query to get the specific information that you seek from the map.

\textbf{2.3.3 \textit{jMap Syntax and Process}}

The syntax of \textit{jMaps} is based on the Relationship Oriented paradigm, or on relating sets and set members. In \textit{jMaps} the relationships are represented by (vertical) tuples/columns. The \textit{kTuple} (knowledge tuple) construct is the fundamental structure defined by the concepts and instances related by roles.

The relating mechanism is implemented by allocating roles to sets in schema and their instance to set members/instances in map. Compared to diagrams, maps are very compact, offering a rich context within limited space of a computer screen. Maps are created or edited within an organized electronic page – spreadsheet which assures efficient manipulation of relationships (columns) and heavy reuse of components (row).
Figure 7  Diagrams Defined by Map Patterns demonstrates associations of descriptor strings to arcs and nodes. The character "f" ("t") associate the strings to the "tail" ("head") of an arc. The character "m" signifies that the string is attached to the "body" of arc or node. Clustering of arcs - and connected nodes - into graphs is shown by tagging columns with character "&". Graphs are connected if they share at least one node. As is illustrated by graph (A) and (B), reordering of columns and/or rows is an information-preservation operation, i.e. the shape of the graph might change but not the meaning. Descriptors of arcs and nodes are set members. Sets are identified by \{<set name>\} and are defined by enumeration. The schema-level view of a map is obtained first by hiding set instances and then by hiding redundant columns (Figure 7). The schema provides information about joined maps (iMaps) structure and size.

<table>
<thead>
<tr>
<th>A</th>
<th>A</th>
<th>A</th>
<th>3</th>
<th>1 (Graph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>&amp;</td>
<td>&amp;</td>
<td>3</td>
<td>(A)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M</th>
<th>M</th>
<th>M</th>
<th>3</th>
<th>3 (Arc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>1</td>
<td>string4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>1</td>
<td>string5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>1</td>
<td>string6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F</th>
<th>F</th>
<th>F</th>
<th>3</th>
<th>3 (Node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>f</td>
<td>2</td>
<td>string1</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>3</td>
<td>string2</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>2</td>
<td>string3</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7  Diagrams Defined by Map Patterns

For the diagrams on the right side, we have the map as shown on the left side of figure. The map contains three sets namely \{Graph\}, \{Arc\}, and \{Node\}. There are three roles namely 'A', 'M' and 'F' and four instance roles namely '&', 'm', 'f' and 't'. Role 'A' was
allocated to \text{Graph} to allow clustering of columns (i.e. relationships of instances) with instance role '\&'. Role 'M' was allocated to \text{Arc} to allow allocating of instance role 'm' to the instances 'string4', 'string5' and 'string6'. Role 'F' was allocated to \text{Node} to allow allocating the instance roles 'f' and 't' to the members/instance of these sets.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{map_schema.png}
\caption{Schema view of Map}
\end{figure}

Large \textit{jMaps} models can be viewed clearer by using group function in Excel to hide the irrelevant columns and rows.

In general, abstract concepts appear on the right of the map in bold and between curly brackets. They can be thought of as a heading of a table column or a row. The instances would then be the actual contents listed in the table. Each column is to be read vertically using the syntax that was described above. For every “variable” you come across when reading down a column, you must read across towards the right of the map, to see which concept or instance the variable is referring to. Beside each instance, and under the total number of instances, represents the number of times the instance is referred to.

\subsection{2.3.4 \textit{jMap} Notation}

\textit{jMap} notation can addresses many topics such as following\cite{9, 10, 12}:

\begin{itemize}
  \item Information system architecture
\end{itemize}
• Recovery and reuse of system patterns
• Evolving information systems
• Software evaluation and renewal
• Systems workstations
• Automation of system design
• Modeling of web sites and knowledge hubs

JMap notation syntax was listed and explained as follows:

• **Generic JMap concepts could be one of the following:**

  A - Template Aggregation
  T – Template
  Y – Dominant
  Z – Descriptive
  K – Identifier
  O – Identity
  H – Hierarchy
  I - Generalization - "parent" or "heir"
  P - Aggregation - "whole" or "part"
  U - Uses or used
  D - Dependence
  S - Sequence - position in a sequence
  F - Flow "from" or "to"
  L - Flow "from", "to" and "loop"
  X - Unique Qualifier
  M - Association
  G - Guard or Goal
  E - Event
V - Value
? - User defined

- **Instances in jMap can be mapped to concept by using following symbols:**

1 … * - identifier or value
o - column marker
h - tree root
l … * - branch
f - from:
t - to:
b - both
m - many or middle:
d - destination:
s - source:
l - loop
a - assertion
e - exception
x - unique row marker
v - related
c - composite
t - true
f - false
o - otherwise
t - implied true
f - implied false
- e - enabled
d - disabled
? - user defined
u - update
2.3.5 Representing Models in jMap Notation

jMap notation is a formal notation, which can represent various systems and processes, and is equivalent to any other notations. Its distinctive feature from other notations is its P2S properties (Processable, Plugable and Searchable) in addition to its excellent visual presentation. JMap notation is also a complete notation and is equivalent to other notations in all aspects. The following subsections describe examples of JMap models which directly converted from other notation models: Entity-Relationship model and UML model.

2.3.5.1 Representing Entity Relationship Model

As an example, we represent a Customer-Orders ER model in JMap notation. In this Customer-Orders Entity-Relation model, we describe the following relational schema:

Entity: Customer { Customer Number, Name, Telephone No, Credit Limit, Balance}

Order { Order Number, Item, Quantity}

Relationship: customers places orders {Customer No, Order No, Date}

To construct a JMap for this ER relationship, the following has to be performed:

1) Identify component types i.e. identification of sets by name.

2) Enumerate sets and identify connector types

3) Create connectivity columns/map by `connecting` components with characters “f” and “t”.

4) Use “M” to identify association. Enhance connectivity columns by using characters “m” to represent association between the attributes
5) Use characters "v" to stress uniqueness/identity of an entity.

6) Use characters "F" to identify columnwise for the sets with members connected by "f" or "t".

7) To create schema view by first hiding set members and then hiding redundant columns.

This simple ER relationship was presented in Figure 9 Customer and Orders Model Represented by jMap. In this figure, we can see the attributes of the entities are represented in the right most column as concepts, their relationship were mapped in the left columns (0-7). Column 8 is the counter of the relations relevant to this concept, column 9 is the total number of instances of the concepts. They (column 8, 9) are equivalent to summary fields in ER relationship. Obviously, the information in the above ER model was completely represented by this jMap.
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>t 2</td>
</tr>
<tr>
<td>f</td>
<td>t 2</td>
</tr>
<tr>
<td>f</td>
<td>1 t 3</td>
</tr>
<tr>
<td>f</td>
<td>1 t 1</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>1</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 9** Customer and Orders Model Represented by jMap
2.3.5.2 Representing UML Models in jMap Notation

The following jMap diagram Figure 10  A Web Design jMap Model Converted from Figure 6 UML model shows a complete jMap notation model directly converted from Figure 6. We define the following jMap schema to represent the contents from the diagram:

Table 5 A jMap Schema for Representing Web Designs

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Concept</th>
<th>UML Equivalency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{Content}</td>
<td>Description of the jmap content</td>
</tr>
<tr>
<td>X</td>
<td>{Edge Type}</td>
<td>Aggregations Relationships (Generalization and composition)</td>
</tr>
<tr>
<td>E</td>
<td>{Edge Component}</td>
<td>Association relationships(dependency, associations)</td>
</tr>
<tr>
<td>F</td>
<td>{Node Name}</td>
<td>Class Entity</td>
</tr>
<tr>
<td>F</td>
<td>{Node Color}</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>{Multiplicity}</td>
<td>Relationship multiplicity</td>
</tr>
<tr>
<td>A</td>
<td>{Content Source}</td>
<td>The source of this information for tracebility</td>
</tr>
<tr>
<td>A</td>
<td>{Author}</td>
<td>The authors creating this jMap for reference</td>
</tr>
</tbody>
</table>

Figure 10 is constructed by using above jMap schema. As seen from this jMap, the visual graphic mosaic from UML diagram was converted to text-based fields and relationship maps on a Excel spreadsheet. All the information residing in the UML diagram for the web design diagram in Figure 6 was represented completely on the jMap. It is easy for human to understand a graphic notation such as UML, however, the graphic notations are not easily processable and plugable. Clearly, this jMap representation renders the model of capability to be plugable in other analysis applications, whole or parts.
2.4 The Analysis of jMap Web Model

As exampled in previous sessions, jMap notation is unique in its excellent processability and convenience. It is one of the best notations for representing the retrieved information model as from web sites. Web mining can be easily done on these processable models. Currently, there are number of separate projects ongoing to process jMap model known as “Context+” projects.
3. Web Mining With jMaps

3.1 Website Structure Graph Model to jMap Model Mapping

UML model is used for web site forward engineering. It is designed as a visual notation for communicating design ideas, however its processability is quite complicated and limited. The web hierarchy model is an intuitive and well-understood model for web forward and reverse engineering and implemented by most websites in reality. jMap notation is suitable for representing the reverse-engineered website information. Its excellent processability and easy of use and variety of data analysis tools make it the best fit for web mining notation. jMap technology is a well-structured web site container, which can represent a web site contents, structure information.

In this report, the research and software development project deals with two aspects of web mining: information retrieval and jMap conversion, which covers both web site content mining and web site structure mining. However, the content and structure analysis part is not in current project scope. For more information, jMap processing utilities such as “Context+” can be referenced.

3.2 Website Structure Recovering using BFS

3.2.1 Breadth-First-Search Algorithm (BFS)

Breadth-First Search(BFS) is one of the simplest algorithms for searching a graph and the archetype for many important graph algorithms such as Prim’s minimum-spanning-tree algorithm and Dijkstra’s single-source shortest-paths algorithm[16].

30
Given a graph $G = (V, E)$ and a distinguished source vertex $s$, BFS systematically explores the edges of $G$ to discover every vertex that is reachable from $s$. It computes the distance (smallest number of edges) from $s$ to each reachable vertex. It also produces a BFS tree with root $s$ that contains all reachable vertices. For any vertex $v$ reachable from $s$, the path in BFS tree from $s$ to $v$ corresponds to the shortest-path from $s$ to $v$ in $G$, that is, a path containing the smallest number of edges. This algorithm works on both directed and undirected graphs.

BFS is named because it expands the frontier between discovered and undiscovered vertices uniformly across the breadth of the frontier. Namely, BFS algorithm discovers all vertices at distance $k$ form $s$ before discovering any vertices at distance $k+1$.

Breadth-First-Search results in a BFS tree. Its root is the source vertex $s$. It starts at scanning $s$’s adjacency list to explore new vertices connected to $s$ and saves the discovered vertices in the queue. Whenever a white vertex $v$ is discovered in the course of scanning the adjacency list of an already discovered vertex $u$, the vertex $v$ and the edge $(u,v)$ are added to the tree. $u$ is called the predecessor or parent of $v$ in the BFS tree. Since a vertex is discovered at most once, it has at most one parent.

As BFS tree only preserves a small portion of website structure information in the tree because one vertex is only discovered at most once, it is used mainly as the exploration algorithm for the "Website jMap Builder" program to discover pages in the site. It saves
the discovered new page hyperlinks in the queue for the program to retrieve and analyze information from the pages. The relationship between the retrieved pages were analyzed using various string sorting and searching algorithms. Since sorting and searching algorithms are trivial program tasks, they will be ignored in detail from this report.

Algorithm

BFS algorithm can be presented as in the following peseudo code:

BFS(G, s)
For each vertex u ∈ V[G] – {s}
        do color[u] ← WHITE
        d[u] ← ∞
        π [u] ← NIL
        Q ← ∅
        Enqueue(Q, s)
While Q != ∅
        Do u ← Dequeue(Q)
        For each v ∈ Adj[u]
                Do if color[u] ← GRAY
                D[v] ← d[u] +1
                π [u] ← u
                Enqueue(Q, v)
        Color[u] ← BLACK

Analysis

The total run time for a BFS-based algorithm is O(V+E), i.e., the running time is proportional to the total number of nodes and total number of vertices in the graph. When BFS algorithm is used to retrieve information from a web site, since the number of pages in a web site is unknown and the time spent on the retrieving pages is much more than the analysis on the local machine. therefore the total running time is linear, O(V), i.e., it is only related to the number of pages retrieved. The actual speed is totally depends on the speed of internet connections and the number of pages will be analysed.
3.2.2 *Generic Website jMap Schema*

jMap technology is flexible and complete for representation of a web site's contents and structure information. This project only uses a simple set of web page's attributes in the jMap schema. The information retrieval from a web page is currently limited to of the following concepts in the schema as described below. This schema could be easily extended with minor change of the design of the program.

- {HTML/TEXT}
- {QUERY}
- {FTP}
- {MAILTO}
- {IMAGE}
- {TITLE}
- {Size}
- {TOKEN}
- {LEVEL}
- {BOOKMARK}
- {LINKOUT}
Figure 11 Generic jMaps Template for Website Model

3.2.3 Mapping Website Attributes and BFS tree attributes to jMap Concepts

BFS algorithm forms the basis of the software design. However, it is only used as an exploration method. the methods used for retrieving and analysing web page attributes are mostly independent from BFS algorithm. Various techniques and program libraries were employed to extract and analyse attributes from each HTML page. The mapping between web site attributes and jMap attributes are computation intensive. The relationship between HTML pages was the result of analysis by means of sorting and searching keys. It was reflected on jMap relationship mapping fields as identified by "t" and "f" symbols. This relationship is considered to be complete and honest to the original web organizations as identified in generic jMap templates.
Table 6  Website Attribute Mapping To jMap Concepts

<table>
<thead>
<tr>
<th></th>
<th>jMap Schema attribute</th>
<th>HTML Page Attribute</th>
<th>Web Graph Attribute</th>
<th>BFS Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{ LEVEL }</td>
<td></td>
<td>explored BFS tree depth</td>
<td>Level on the tree</td>
</tr>
<tr>
<td>2</td>
<td>{HTML/TEXT}</td>
<td>HTML/TEXT hyperlinks</td>
<td>directed edge from this page as Source vertex with label &quot;hyperlink&quot;</td>
<td>Children of this page</td>
</tr>
<tr>
<td>3</td>
<td>{ MAILTO }</td>
<td>embedded email hyperlinks</td>
<td>discarded edge</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>{ FTP }</td>
<td>ftp hyperlinks</td>
<td>directed edge from this page as source vertex with label &quot;ftp&quot;</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>{ QUERY }</td>
<td>GET and POST query links</td>
<td>directed edge from this page as source vertex with label &quot;query&quot;</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>{ TITLE }</td>
<td>Title of this page</td>
<td>contents of the vertex</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>{ SIZE }</td>
<td>Size of this page</td>
<td>contents of the vertex</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>{ TOKEN }</td>
<td>Context Text of the hyperlinks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>{BOOKMARK}</td>
<td>Total hyperlink to this pages</td>
<td>self-directed edge</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>{ LINKOUT }</td>
<td>Total hyperlink to other pages</td>
<td>children</td>
<td>Children</td>
</tr>
<tr>
<td>11</td>
<td>Hyperlink Relationships</td>
<td>hyperlinks to predecessor can be mapped as back edges, to sibling page as cross edges, to self as self-directed edges, to child page as forward edges. All edges in jMap starts from source vertex with &quot;f&quot; and ends with target vertex with &quot;to&quot;. multiple edges are marked with its cardinality such as &quot;3t&quot;.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2.4 A Website Model Recovered with jMaps

Since a web site jMap model is quite large - in Figure 12 A Tailored Version of Web Site Model for www.cs.concordia.ca. - we show only a small fraction of instances for each concept in the jMap. This figure shows a formatted jMap in Excel spreadsheet. The color scheme was added with jMap processing utilities. For a complete view of this jMap model, an Excel spreadsheet is attached with this report in Appendix B.
3.3 Development Tools

The development tool used is listed as follows:

- PERL, with libraries: PERL/TK8.00, LWP, HTML, MIME, MAIL modules.

  PERL is an open source language. There are plenty of libraries available for download free of charge and license restrictions. PERL is also famous for its text processing, system and network administration capabilities.

- Platform: Windows/UNIX (Solaris)/Linux

  Website jMap Builder software is designed to run on any platform, which a PERL portal is available for it.

- Microsoft Excel

  Excel is a spreadsheet that allows users to organize data, do complicated calculations, make decisions, draw graph from data, develop professional-looking...
reports, convert Excel files for use on the database, access the database, etc. Visual Basic Application (VBA) for Excel provides us with the means to accomplish a wide range of the programing tasks. With VBA, we can create full-fledged custom applications in Microsoft Excel.

3.4 Website jMap Builder Design Constraints

This software is mainly constrained by the power of the computer and the speed of Internet connection. This software is also limited by the following third party software:

- **Microsoft Excel**
  
  Which currently limit the maximum number of 256 column and 65535 rows.

- **PERL**
  
  *Perl* is a open source software. It has abundant modules available for download free of charge. This software makes use of the following PERL modules:
  
  - PERL/Tk for GUI.
  - LWP for accessing to WWW including HTTP, FTP protocols.
  - HTML for extracting links, parsing and analyzing HTML page for page properties.

  However, all the links not in HTML scope such as in java script, flash, image maps are not processable by current libraries.
4. Software Specifications

4.1 Architecture

This software was designed for all platforms with a PERL portal. The back-end program and front-end user interface were developed in PERL and can run on various platforms. Because of the nature of jMap application, the final jMaps have to transform to a PC environment and formatted in Excel spreadsheet.

The main thrust for jMap processing is designed for three front end user interfaces as in Figure 13 Web jMap Builder Architecture:

Figure 13 Web jMap Builder Architecture

A: Email Request Server; B: Local Terminal GUI; C: Web CGI Interface
Configuration A is a service based configuration and designed for heavy load jMapping of big web sites, which need huge amount of CPU power and memory. Ideally, the jMap Website Builder server should be an UNIX server. The user can place a request by sending a formatted email to the mail server. The jMap server will poll the mail server regularly and process the user requests and send results back to the user via email. This is the most powerful way of using this software.

Configuration B is designed for individual user to run this application from his personal computer with fast internet connections. It is most convenient way of using this software.

The Web CGI interface is currently not implemented in current project.

4.2 Functionalities

One of the main features of this project is to develop an automatic tool to explore the given websites and harvest the information from the websites and export it to Excel spreadsheet for processing by jMap Macros and VBA utilities. The main system's functions as seen by the user can be summarized as:

1) To provide user interfaces to specify parameters for web site jMapping.

2) To explore the websites and export the result.

3) To format the results to a proper jMap notation in Microsoft Excel spreadsheet.

4) Cross platform capability and scalability.
4.3 Software Design

4.3.1 Architecture Design

This software targets cross-platform application and scalability. The architecture design for interprocess communications was limited to only plain text files. Although Figure 13 Web jMap Builder Architecture shows three user interfaces in the architecture, currently only two user interfaces were implemented so far as shown in Figure 14 Software Architecture Design Diagram.

The first design is to use an Excel spreadsheet GUI interface and VBA macros to explore and process jMaps in a local environment. The GUI helps to gather user requirements and invokes the main thrust program to process website jmaps. The user can open and format the jMap results through Excel GUI.

The second design aims to use a more powerful server computer which can process much larger sites and is capable of cross-site jointed mapping. A user has to send a formatted email to a designated email address to request the Website jMap Building service. The server will process the requests and email the results back to the user in the attachments.
4.3.2 Detailed Design

4.3.2.1 Main Thrust Program

The main thrust program (file: webjmapm.pl) implements the BFS algorithms to explore the internal pages in the given web site and uses JFILE and JMAP classes to retrieve and process information from the discovered pages. The design details are described as below in Figure 15. Website jMap Builder Library Classes. These two perl classes are object-oriented PERL packages and used by the main thrust program. They can be installed in PERL lib directories as well as the main program working directory.
Figure 15  Website jMap Builder Library Classes

**JMAP**

This class is used for process jMaps.

**Protected Properties:**

- **JFILES : JFILE = 0**
  
  An array of JFILES.

- **MAX_LEV : integer = 0**
  
  The maximum level of the BFS tree to be build.

- **TYPES : hash = 0**
  
  A hash of different file types.

- **HEADERS : Hash = 0**
A hash of headers.

`URL_CNT` : Integer array = 0

An array of counters for different type of URLs.

`LINKS` : An array of URLs

URLs explored.

`LOGFILE` : string = ""

Log file name and path.

**Public Methods:**

new () : &JMAP

Constructor.

`limit_jfile()` : void

In case of too many jfiles, we eliminate the outside links or links without properties.

`order_url()` : void

Sort and put `LINKS` in a specified order.

`get_dist_vals()` :

process all distinct values from jfile objects return a list with distinct values:

`set_val()` :

Mutator.

`get_val()` :

Accessor.

`get_header()` :
Map a concept to proper header for jMap.

is_equal () :

Compare two values.

jmap_surl () :

Map different type of URL separately to jMap.

jmap_hz () :

Joined mapping the JFILEs horizontally.

jmap_url () :

joined mapping all URLs.

jmapping_token () :

Joined mapping tokens.

jmapping () :

Joined map all concepts to jMap.

get_title () :

Obtain a title for this token or page.

find_size () :

Return the size of a page.

logging () :

Log messages to a file for display.

JFILE

This class is used for storing properties of a HTML page.

Protected Properties:

URL : string = ""
The URL of this web page.

LEVEL : integer = 0

The BFS search tree level, i.e., the depth of the BFS tree.

TITLE : string = 0

The title of a web page.

SIZE : integer = 0

The size of a page which only the text in HTML format was counted. The linked files were not included.

BOOKMARK : integer = 0

The number of bookmarks in this HTML page.

LINKOUT : integer = 0

The number of links to other pages in this page.

LINKIMG : integer = 0

The number of images in this page.

LINKS : array of string = ""

All the hyperlinks found in this page.

LAST_MODIFIED : Time = 0

The last modified timestamp of this page.

RS_CODE : integer = 0

HTTP request return code.

RS_TIME : integer = 0

The number of seconds of this HTTP request.

TOKENS : an array of strings = ""
All the context tokens found in this page.

**Public Methods:**

`new()`: &JFILE

Constructor of a JFILE.

`set_val()`: argtype = default

Mutator to set class attributes.

`sort_links()`: 

Sort LINKS in alphabetical order.

`sort_token()`: 

Sort TOKEN in alphabetical order.

`is_linked()`: 

Test if a link contained in this page's hyperlinks.

`get_TokenFoundCnt()`: 

Get the count of token found in this page.

`get_val()`: 

Accessor to obtain class attribute value.

4.3.2.2 Mail Order Server

The mail order server (pop3svr.pl) is based on POP3 protocol. It acts as a POP3 client polling a mail server regularly and parse messages to save in commands.txt file. If there are request, it calls the main thrust program to process the request. The main thrust program (webmapm.pl) will process the request and send the result back to the user via email.
4.3.2.3 Main Excel Macros

There are three Excel Macros were created to attach to the buttons on Excel GUI.

sub initEnv():

This macro initializes some global variables.

sub RunPerl():

This macro is used to invoke PERL/TK GUI for the main thrust program and attached to "Run Website JMAP GUI" button in WebjMap sheet of file jMap.xls.

Sub OpenJmapReopt

This macro is used to open and format JmapReport.txt file.

Sub openJmap

This macro is used to open and format jmap.xxx.txt temporary jMap files produced by the main thrust program.
5. Software User Manual

5.1 System Requirements

Before running this program, you need check if your system meets following requirements:

**Hardware Requirements:**

- CPU: Intel Pentium 200 or faster /Sun Workstation/HP Workstation
- Monitor – SVGA (800x600) or better
- RAM – at least 256 MB
- Pointing device: Mouse.
- Hard Disk Space: at least 50 MB free space for PERL/Tk/Website JMAP installation.

**Software:**

- Microsoft Excel 97 or Excel 2000
- PERL 5.005 or later with Required libraries as in Section 5.2.2 PERL Libraries

**Platform:**

The program was tested to run on the following platforms:

- Windows 95/98/Me/2000/NT4
- RedHat Linux
• UNIX (Sun Solaris)

5.2 Software Installation

5.2.1 PERL 5.005

Perl 5.005 or later can be downloaded free from the following sites:

• For Win32 based platform, http://www.activperl.com

• For Solaris/Linux based platform: http://www.cpan.com

The installation procedure was described in those websites and accompanying with the software.

5.2.2 PERL Libraries

Perl library modules can be downloaded from previously mentioned websites free of charge and can be installed according to the installation procedure as specified in the readme file accompanying with the software.

• PERL/TK 8.00 or later (For GUI. optional for Server)

• LWP (mandatory)

• HTML (mandatory)

• Mail: Mail::Sendmail (mandatory for Server. optional for GUI application)

5.2.3 Microsoft Excel

• Microsoft Excel 97 or 2000
5.2.4 Website jMap Builder Software

5.2.4.1 MAP Home Directory

To install "Website jMap Builder", the first step is to create a directory to host the directory structure and files of jMap Builder. This directory is called JMAP_HOME. It is recommended that the name of this directory should contain the version number, so multiple versions can co-exist in one system. For example, D:\JMAP5.

5.2.4.2 Directory Structure and Files

Web Site jMap Builder software has the following directory structure and files:

Table 7 Web Site jMap Builder Directory Structure and Files

<table>
<thead>
<tr>
<th>Directory</th>
<th>Files</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$JMAP_HOME/bin</td>
<td>gui</td>
<td>UNIX shell script to start GUI in UNIX shells.</td>
</tr>
<tr>
<td></td>
<td>gui.bat</td>
<td>DOS shell script to start GUI in DOS shell.</td>
</tr>
<tr>
<td></td>
<td>gui.pl</td>
<td>PERL/TK GUI program.</td>
</tr>
<tr>
<td></td>
<td>jmap</td>
<td>UNIX shell script to start GUI in UNIX shells.</td>
</tr>
<tr>
<td></td>
<td>jmap.bat</td>
<td>DOS shell script to start GUI in DOS shell.</td>
</tr>
<tr>
<td></td>
<td>JMAP.pm</td>
<td>User PERL class library for jMap processing.</td>
</tr>
<tr>
<td></td>
<td>JFILE.pm</td>
<td>User PERL class library for page information storage</td>
</tr>
<tr>
<td></td>
<td>jMap.xls</td>
<td>Excel template and Macros for jMap formatting.</td>
</tr>
<tr>
<td></td>
<td>pop3svr.pl</td>
<td>POP3 mail request server program.</td>
</tr>
<tr>
<td></td>
<td>webjmap4.pl</td>
<td>Generic version of webjmap building program.</td>
</tr>
<tr>
<td></td>
<td>webjmapc.pl</td>
<td>Web jMap building program for CGI application.</td>
</tr>
<tr>
<td></td>
<td>webjmapg.pl</td>
<td>Web jMap building program for GUI application.</td>
</tr>
<tr>
<td></td>
<td>webjmapm.pl</td>
<td>Web jMap building program for mail server application.</td>
</tr>
<tr>
<td>$JMAP_HOME/conf</td>
<td>command.txt</td>
<td>An interface file between GUI-WebjMapx program or pop3svr.pl and webjmapx.pl</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>session.txt</td>
<td>A persistent storage of session number. The number is incremental.</td>
</tr>
<tr>
<td></td>
<td>jMap.conf</td>
<td>Website jMap Builder configuration file.</td>
</tr>
<tr>
<td></td>
<td>stop.txt</td>
<td>An user control interface file between GUI-WebjMapx program or pop3svr.pl and webjmapx.pl</td>
</tr>
<tr>
<td>$JMAP_HOME/doc</td>
<td>documentation files</td>
<td>documentation of this software</td>
</tr>
<tr>
<td>$JMAP_HOME/jmaps</td>
<td>jmap.xxxxx.txt</td>
<td>Result jMap in text format. xxxx represent the session number.</td>
</tr>
<tr>
<td></td>
<td>jmap.cxxxxx.txt</td>
<td>cross-sites joint mapping text result.</td>
</tr>
<tr>
<td></td>
<td>JmapReport.txt</td>
<td>a summary report for last sessions.</td>
</tr>
<tr>
<td>$JMAP_HOME/lib</td>
<td>*</td>
<td>Perl libraries</td>
</tr>
<tr>
<td>$JMAP_HOME/lib/activeperl</td>
<td>*</td>
<td>for WIN32 portals. From ActivePerl.</td>
</tr>
<tr>
<td>$JMAP_HOME/lib/CPAN</td>
<td>*</td>
<td>For UNIX/LINUX (also good for WIN32) from CPAN.</td>
</tr>
<tr>
<td>$JMAP_HOME/log/</td>
<td>pop3svr.log</td>
<td>pop3 server log file.</td>
</tr>
<tr>
<td>$JMAP_HOME/tmp/</td>
<td>jmap.log</td>
<td>temporary log file acting interfaces between GUI and webjmapx.pl program.</td>
</tr>
<tr>
<td></td>
<td>temp.html</td>
<td>temporary web page used for parsing.</td>
</tr>
</tbody>
</table>

* some libraries may be not distributed with this software.

### 5.3 Local Terminal Application

#### 5.3.1 Run Local Terminal Application on a PC

Local terminal application (LTA) is the most convenient way to map a web site to its jMap notation. The following steps are recommended to use Website JMap Builder on a PC from Excel spreadsheet interface.

1: Open Microsoft Excel, open **jMap.xls** in $JMAP\_HOME/bin directory, click "Enable Macros" button to open the file when Excel prompts the dialogue. The open Sheet should look like Figure 16 MS Excel Macros Interface. If you select
**Disable Macros** button, then you will be unable to run the Macros in the program.

**Website jMap Builder 4**

Copyright © Gen-Strategies.com 2000-2005

<table>
<thead>
<tr>
<th>Web jMap Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JMAP HOME Directory</strong></td>
</tr>
<tr>
<td><strong>PERL Compiler Path</strong></td>
</tr>
<tr>
<td><strong>This workbook Name</strong></td>
</tr>
</tbody>
</table>

Please do not shift above cells

Run Website jMap GUI  Open Last Report  Open JMAPs

Figure 16  MS Excel Macros Interface Worksheet

2: Enter configuration information in the Cells B7:B9 or (R7C2 : R9C2). It is mandatory to enter $\text{JMAP\_HOME}$ directory path (post-fixed with "\") to B7. TO start Website jMap Builder GUI, click "Run Website jMap GUI" button. A GUI looks like Figure 17 "Website jMap Builder" GUI should appear. If not, check the $\text{JMAP\_HOME}$ is correct or not.
Figure 17 "Website jMap Builder" GUI

3: Enter root URL of the websites, maximum level of the site to be explored, maximum number of URL to be retrieved, check the option to indicate this root to be in the cross-site mapping or not, click "Add" to add this root to the batch request. In case of error, click the entries in the list box, click "delete" to remove them.

4: Click "Start" button to process the request. Now the GUI text area will display the processing information. The program can be stopped by clicking "Stop" button. After all processing is accomplished, the text area should have displayed the information and the "Stop" button should become "View Report" button. Click the "View Report" button, a summary report will be displayed on the text area.
5.3.2 Run Local Terminal Application on UNIX/Linux

To run "Website jMap Builder" On UNIX or Linux system, the Microsoft Excel GUI is not available on these systems. Instead of starting from Excel, goto $JMAP_HOME/bin directory, type "gui" or "jmap" at the command line. Then, follow the steps in Section 5.3.1 Run Local Terminal Application on a PC (step 3 and 4).

5.4 POP3 jMap Builder Server for Processing Mail Requests

5.4.1 Single Process Configuration

The following instructions are given for setting-up a POP3 server to process requests and email back jMaps to the users.

1: Open an email account on a mail server which supports POP3 protocol. Note down the server name, account name, and password.

2: Use a text edit to open $JMAP_HOME/conf/jMap.conf. and set your configurations. The following shows an example configuration. The "OS" is the operating system on which pop3svr.pl runs, not your POP3 mail server's OS. "Outgoing SMTP Server" is your default mail server to send emails which support SMTP. "Reply Email address" is your email address. "Request Subject" is a filter the pop3svr.pl program will look for requests. It is useful especially when this mail account is not just used for jMap requesting service.

```
#jMap.conf

#jMAP configurations. it must have spaces before and after "=" sign

#OS: UNIX or WIN32
OS = WIN32
```
#email host configuration.

Outgoing SMTP Server = smtp.server.ip.or.name

Reply Email address = your.name@domain.ca

#JMAP POP3 Configuration

POP3 Server = mail.domain.com

POP3 User = xxxxx

POP3 Password = xxxxxx

Request Subject = Website jMap Building Request

3: Run pop3svr.pl from $JMAP_HOME/bin. Up to now, the server is up and running and ready to serve the requests.

5.4.2 Multiple Process Configuration

Multiple POP3 Servers can be run on different platform and different architecture. It is recommended to run multiple processes on a single server or different servers to suffice the service reliability and increase throughput.

1: To run multiple pop3svr.pl process on a single server environment, it is highly recommended to duplicate the $JMAP_HOME directory structure and run different processes from their own home directories to completely segregate the server processes. Currently, there is no synchronization mechanisms build-in the server process to control the accesses to the shared resources. It is likely to clash when running multiple processes from the same location. A recommended structure looks like this:
2: To run multiple pop3svr.pl process on multiple server environment, it is also highly recommended to run multiple processes on each server and all processes should be completely segregated as stated in previous configuration (1).

### 5.4.3 Process Configuration

The jMap Builder Mailer and Server does not work without specifying configurations in the $JMAP_HOME/conf/jMap.conf file. The file looks like the follows.

```plaintext
#jMAP configurations, it must have spaces before and after "=" sign
#OS: UNIX or WIN32
OS = WIN32
#email host configuration
Outgoing SMTP Server = smtp.xxx.ca
Reply Email address = your.email@address.ca

#JMAP POP3 Configuration
POP3 Server = xxx.ddd.200.20
POP3 User = pupeng
POP3 Password = xxxxxx
Request Subject = Website jMap
Polling Interval = 10 minutes
```
Outgoing SMTP server: the server you normally send out MIME type emails in your local LAN or dialup networks.

Reply Email address: The email address you want the user to know.

Pop3 Server: The server on which you have the email account setup for service. If DNS is not available on the network, an IP address is needed.

Pop3 User: your account name on the POP3 Server used for request service.

POP3 password: password to access this POP3 mail account.

Request Subject: a filter for the server program to select proper mail for parsing for request. This is useful for separating the request from your personal mails.

Polling Interval: the time interval between this server checks the POP3 emails. Depends on the traffic of the request and the number of processes. Reducing the interval allows the server to process more requests.

5.4.4 Sending Email to Request Website jMap Building Service

To request a service from the server, a user has to send an email to the designated address with the following:

1: With the configured key words in the subject line.

2: Commands in the text body using format:

http://your.domain.com <spaces or tab> max_level <spaces or tab> max_url <spaces or tab> jmap_style <spaces or tab> in_group

max_level: the maximum depth of BFS tree you want to explore this site.
Max_url: the maximum number of urls you want to retrieve and analysis.

jmap_style: 0 for no token string analysis, 1 for jMaps including Token analysis.

In_group: 0 for individual website analysis, 1 for cross-site jMapping.

3: Multiple command lines were allowed in the mail.

5.5 Load Web Site jMap Building Report

If the processing of website jMap was done on a UNIX/Linux server, the temporary jMap files and the JmapReport.txt should be transferred to a PC based system and the files should be saved in $JMAP_HOME/jmaps/ directory.

To load the JmapReport, simply click the "Open Last Report" button. The report should be shown in a separate sheet looking like Figure 18 Open Web Site jMap Building Report.
5.6 Formatting Text jMap to Excel Spreadsheet

1. To format a text based jMap temporary file to Excel spreadsheet, all jMap files have to be moved to a PC and the jmap files in $JMAP_HOME/jmaps/ have to copied to $JMAP_HOME/jmaps/ in a PC installation.

2. Then, click the "Open jMAPs" button. input JMAP session File dialogue will be prompted: Figure 19 Enter Session Number to Open jMap.
3. Enter the jMap session number. If it is a cross-site join mapping file, enter c+session number. Click "OK" button to load the jMap file and format it. The formatted result looks like Figure 20 Formatting jMap Results.

Figure 19 Enter Session Number to Open jMap

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 20 Formatting jMap Results
5.7 Web Mining Process with Context+ Utilities

The formatted jMaps can be easily processed by Excel Macros developed previously as "Context+". The use of these programs is beyond the scope of this report.
6. Conclusion And Future Work

6.1 Summary and Conclusions

In brief, we summarize the following as the work carried out in this project:

1) A BFS based main thrust program was developed to automatically retrieve and analyze a web site's contents and structure information.

2) Web page organization in a website was viewed as a hierarchy graph. and BFS algorithm was used for discovering pages in the site.

3) A website jMap generic template was developed and implemented in the software.

4) jMap representation of a website contents are achieved by the developed software and shown to be well suited for web mining. jMaps syntax is simple and robust. All discovered contents by the program was transformed to modular, possible jMap notation. And the main jMap result can be readily formatted to processable jMaps. jMap notation accommodated all retrieved information.

5) A GUI for local terminal application was developed and tested.

6) A POP3 server for mail request service for website jMap building was developed.

7) The formatted jMaps can be processed by any standard jMap utilities such as "Context+".

We conclude that jmaps provide a viable alternative to the design and analysis of web-based information.
6.2 Future Works

This project can be extended in the following ways:

1) A web CGI GUI for the main thrust program.

2) Some algorithms for sorting and searching can be improved to boost performance.

3) To evaluate and improve the usability of GUI is yet another work needs to be done.

4) To increase the current program's capability to process pages containing FRAMESET, .JavaScripts and ASP scripts.
References


21) **Jim Conallen**, “Modeling Web Application with UML”,

22) **Jim Conallen**, “UML Extension for Web Applications Specification 0.91”,
    http://www.conallen.com/technologyCorner/webextension/WebExtension091.htm

23) **Jim Conallen**, “Modeling Web Application Design with UML”:
Appendix A: Source Code

The program was coded in VBA and PERL, the project consists of three parts:

- Excel Macros
- Main thrust PERL program.
- Server Program to process POP3 mail request.

A-1 Excel Macros

A-1.1 Global Variable

' Global variable declaration
Dim JmapHomeDir As String 'JMAP HOME
Dim JmapManualSheetName As String 'JMAP Tmp file DIR
Dim JmapFileDir As String 'JMAP binary DIR
Dim JmapBinDir As String 'JMAP temporary txt file dir

A-1.2 Sub initEnv()

Option Explicit
Sub initEnv()
Sheets("WebjMap").Select
JmapHomeDir = Range("B7").Value

If (JmapHomeDir = "") Then
JmapHomeDir = "d:\jmap\bin\" 'default
End If

JmapFileDir = JmapHomeDir + "conf"
JmapBinDir = JmapHomeDir + "bin\"
JmapTxtDir = JmapHomeDir + "jmaps\"
End Sub

A-1.3 Sub RunPerl()

Sub RunPerl()
Dim GuiBatchFile As String
Workbooks("jMap.xls").Activate
InitEnv
ChDir (JmapBinDir)
GuiBatchFile = JmapBinDir + "gui.bat"
ActiveSheet.Select
Dim dummy As Double
    dummy = Shell(GuiBatchFile, vbHide)
End Sub

A-1.4 Sub OpenJmapReport()

Sub OpenJmapReport()
    'OpenJmapReport Macro
    'Macro recorded 2/11/2002 by Philip Peng
    InitEnv
    Workbooks.OpenText Filename:=JmapTxtDir + "JmapReport.txt", Origin:= _
        xlWindows, StartRow:=1, DataType:=xIDelimited, TextQualifier:= _
        xIDoubleQuote, ConsecutiveDelimiter:=False, Tab:=True, Semicolon:=False, _
        Comma:=False, Space:=False, Other:=False, FieldInfo:=Array(Array(1, 1), _
        Array(2, 1), Array(3, 1))
    Rows("1:1").Select
    Selection.Insert Shift:=xlDown
    Columns("A:A").ColumnWidth = 8
    Columns("B:B").Select
    Selection.ColumnWidth = 30.89
    Columns("C:C").Select
    Selection.ColumnWidth = 35.44
    Rows("1:1").Select
    Selection.Insert Shift:=xlDown
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "Last Website jMap Reports"
    Range("A1:C1").Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .ShrinkToFit = False
        .MergeCells = False
    End With
    Selection.Merge
    With Selection.Font
        .Name = "Arial"
        .Size = 14
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .outlineFont = False
        .Shadow = False
        .Underline = xUUnderlineStyleNone
        .ColorIndex = xIC automatic
    End With
    Selection.Font.Bold = True
    Selection.Font.ColorIndex = 41
    Range("A2").Select
    ActiveCell.FormulaR1C1 = "Session"
Range("B2").Select
ActiveCell.FormulaR1C1 = "Root URL"
Range("C2").Select
ActiveCell.FormulaR1C1 = "jMap Result"
Range("A2:C2").Select
Selection.Font.ColorIndex = 51
Selection.Font.Bold = True
Selection.Font.ColorIndex = 0
Range("A3:C3").Select
With Selection.Font
  .Name = "Arial"
  .Size = 12
  .StrikeThrough = False
  .Superscript = False
  .Subscript = False
  .OutlineFont = False
  .Shadow = False
  .Underline = xlUnderlineStyleNone
  .ColorIndex = xlAutomatic
End With
With Selection.Font
  .Name = "Times New Roman"
  .Size = 12
  .StrikeThrough = False
  .Superscript = False
  .Subscript = False
  .OutlineFont = False
  .Shadow = False
  .Underline = xlUnderlineStyleNone
  .ColorIndex = xlAutomatic
End With
Range("A1").Select
End Sub

A-1.5  Sub OpenJmap()

Sub OpenJmap()

  ' OpenJmap! Macro
  ' Macro recorded 11/27/99 by Richard Peng
  ' ChDir "C:\WINDOWS\TEMP"
  Dim JmapFile As String
  Dim JmapSession As String
  Dim WorkBookName As String
  Dim Prompt, Title As String
  Dim JmapSheetName As String
  Dim Message, boxTitle, Default As String

  InitEnv 'get configurations from the webjMap page

  Message = "Please enter a JMAP session number:" ' Set prompt.
  boxTitle = "Input JMAP Session File" ' Set title.
  Default = "1000" ' Set default.
  ' Display message, title, and default value.
JmapSession = InputBox(Message, boxTitle, Default)

' Use Helpfile and context. The Help button is added automatically.
' MyValue = InputBox(Message, Title, . . . "DEMO.HLP", 10)

' Display dialog box at position 100, 100.
' MyValue = InputBox(Message, Title, Default, 100, 100)

JmapFile = "jmap." + JmapSession + ".txt"
JmapSheetName = "jmap." + JmapSession

Application.ScreenUpdating = False
'setup R1C1 reference style
With Application
    .ReferenceStyle = xlA1
    .xIR1C1
    .UserName = "Philip Peng"
    .StandardFont = "Arial"
    .StandardFontSize = "10"
    .DefaultFilePath = JmapTxtDir
    .EnableSound = False
    .RollZoom = False
End With

WorkBookName = Range("B9").""jMap.xls"
If WorkBookName = "" Then
    WorkBookName = "jMap.xls"
End If

'the following code will open the tempory file and move it to a new sheet

    Windows(JmapFile).Activate
    Sheets(JmapSheetName).Select
    Application.StatusBar = True

' copy loaded data and copy and paste to sheet2. rename to jMap
    Cells.Select
    Selection.Copy

' starting format jMap
' Format_jMap
' move jmap.tmp to c458.xls
    Windows(JmapFile).Activate
    Sheets(JmapSheetName).Select
    Sheets(JmapSheetName).Move After:=Workbooks(WorkBookName).Sheets(1)
    Sheets(JmapSheetName).Select
    Sheets(JmapSheetName).Name = "jMap"
    Cells.Select
    Selection.ColumnWidth = 2
    ActiveWindow.Zoom = 75
With ActiveSheet.Outline
  .AutomaticStyles = False
  .SummaryRow = xlAbove
  .SummaryColumn = xlLeft
End With
Range("A1").Select

' hyperLink Macro
' Macro recorded 03/23/2000 by Philip Peng
'

Dim numJfile As Integer
Dim numLevel As Integer
Dim numURL As Integer
Dim numHtml As Integer
Dim numOther As Integer
Dim numQuery As Integer
Dim numImage As Integer
Dim numMailto As Integer
Dim numFTP As Integer
Dim numTitle As Integer
Dim numSize As Integer
Dim numLinkOut As Integer
Dim numLinkImg As Integer
Dim numToken As Integer
Dim numSkipRow As Integer

Dim link As String
Dim i As Integer
Dim StartRow As Integer
Dim StartCol As Integer
Dim StopRow As Integer

Worksheets("jMap").Activate

numJfile = Range("B1").Value
numLevel = Range("D1").Value
numURL = Range("F1").Value
numHtml = Range("H1").Value
numOther = Range("J1").Value
numQuery = Range("L1").Value
numImage = Range("N1").Value
numMailto = Range("P1").Value
numFTP = Range("R1").Value
numTitle = Range("T1").Value
numToken = Range("V1").Value

numSize = 2
numLinkOut = 2
numLinkImg = 2
numSkipRow = 10

StartRow = 3 + numLevel
StopRow = StartRow + numURL + numTitle + numToken + numSkipRow + 6 'numHtml asked to hyperlink all
StartCol = numJfile + 3
Application.StatusBar = "Formatting JMAP ....."

'format url column width and fonts
Range("A1").Offset(1, StartCol).Select
Selection.ColumnWidth = 60

'title column width
Range("A1").Offset(1, StartCol + 1).Select
Selection.ColumnWidth = 60

'distinguish values column
Range("A1").Offset(1, StartCol - 1).Select
Selection.ColumnWidth = 10
ActiveCell.EntireColumn.Select

Application.CutCopyMode = False
With Selection.Font
  .Name = "Arial"
  .FontSize = "Bold"
  .Size = 10
  .StrikeThrough = False
  .Superscript = False
  .Subscript = False
  .OutlineFont = False
  .Shadow = False
  .Underline = xlUnderlineStyleNone
  .ColorIndex = 40
End With
With Selection.Interior
  .ColorIndex = 35
  .Pattern = xlSolid
  .PatternColorIndex = xlAutomatic
End With

'link in column
Range("A1").Offset(1, StartCol - 2).Select
Selection.ColumnWidth = 10
ActiveCell.EntireColumn.Select

Application.CutCopyMode = False
With Selection.Font
  .Name = "Arial"
  .FontSize = "Bold Italic"
  .Size = 10
  .StrikeThrough = False
  .Superscript = False
  .Subscript = False
  .OutlineFont = False
  .Shadow = False
  .Underline = xlUnderlineStyleNone
.ColorIndex = 46
End With
With Selection.Interior
 .ColorIndex = 35
 .Pattern = xlSolid
 .PatternColorIndex = xlAutomatic
End With

' objects number column
Range("A1").Offset(1, StartCol - 3).Select
Selection.ColumnWidth = 10
ActiveCell.EntireColumn.Select
Application.CutCopyMode = False
With Selection.Font
 .Name = "Arial"
 .FontStyle = "Bold"
 .Size = 10
 .StrikeThrough = False
 .Superscript = False
 .Subscript = False
 .OutlineFont = False
 .Shadow = False
 .Underline = xlUnderlineStyleNone
 .ColorIndex = 46
End With
With Selection.Interior
 .ColorIndex = 35
 .Pattern = xlSolid
 .PatternColorIndex = xlAutomatic
End With

'make clear to read by dim skip points
Dim HTML_Header_SKIP_ROW As Integer
Dim OTHER_Header_SKIP_ROW As Integer
Dim QUERY_Header_SKIP_ROW As Integer
Dim IMAGE_Header_SKIP_ROW As Integer
Dim MAILTO_Header_SKIP_ROW As Integer
Dim TITLE_Header_SKIP_ROW As Integer
Dim FTP_Header_SKIP_ROW As Integer
Dim TOKEN_Header_SKIP_ROW As Integer

' calculate the skip point or range
HTML_Header_SKIP_ROW = StartRow + numHtml
OTHER_Header_SKIP_ROW = StartRow + numHtml + numOther + 1
QUERY_Header_SKIP_ROW = StartRow + numHtml + numOther + numQuery + 2
IMAGE_Header_SKIP_ROW = StartRow + numHtml + numOther + numQuery + numImage + 3
MAILTO_Header_SKIP_ROW = StartRow + numHtml + numOther + numQuery + numImage + numMailto + 4
TITLE_Header_SKIP_ROW = StartRow + numHtml + numOther + numQuery + numImage + numMailto + numFTP + 5
FTP_Header_SKIP_ROW = StartRow + numHtml + numOther + numQuery + numImage + numMailto + numFTP + numTitle + 6
TOKEN_Header_SKIP_ROW = StartRow + numHtml + numOther + numQuery + numImage + numMailto + numFTP + numTitle + numSkipRow + 6

'hyperlinking for url and title
Application.StatusBar = "Setup hypelinks for HTML Pages"
For i = StartRow To StopRow Step 1
  If i = StartRow + numHtml _
    Or i = StartRow + numHtml + numOther + 1 _
    Or i = StartRow + numHtml + numOther + numQuery + 2 _
    Or i = StartRow + numHtml + numOther + numQuery + numImage + 3 _
    Or i = StartRow + numHtml + numOther + numQuery + numImage + numMailto + 4 _
    Or i = StartRow + numHtml + numOther + numQuery + numImage + numMailto + numFTP + 5
    Or i = StartRow + numHtml + numOther + numQuery + numImage + numMailto + numFTP + numTitle + 6 _
    Or (i > StartRow + numHtml + numOther + numQuery + numImage + numMailto + numFTP + numTitle + 6 _
    And i <= StartRow + numHtml + numOther + numQuery + numImage + numMailto + numFTP + numTitle + numSkipRow + 6) _
    Then
      Application.StatusBar = "Skipping the header lines ...."
  Else

    Range("A1").Offset(i, StartCol).Select
    link = ActiveCell.Value
    ActiveSheet.Hyperlinks.Add Anchor:=Selection, Address:=link
    'title link
    Range("A1").Offset(i, StartCol + 1).Select
    'If ActiveCell.Value = "" Then
    'Else
    ActiveSheet.Hyperlinks.Add Anchor:=Selection, Address:=link
    'End If
    'token title hyper link
    If (i > TOKEN_Header_SKIP_ROW) Then
      Range("A1").Offset(i, StartCol + 2).Select
      If ActiveCell.Value = "" Then
        Else
          ActiveSheet.Hyperlinks.Add Anchor:=Selection, Address:=link
          End If
      End If
    End If
End If
Next i

'Selection.Delete Shift:=xlUp
Application.StatusBar = False
Application.ScreenUpdating = True
Range("A1").Select
'tagup R1C1 reference style
With Application
  .ReferenceStyle = xIR1C1
  'xIR1C1
  .UserName = "Philip Peng"
  .StandardFont = "Arial"
  .StandardFontSize = "10"
  .
A-2 JMAP Main Thrust Source Code

A-2.1 JFILE.pm

```perl
#!/usr/bin/perl

package JFILE;

$debug = 0;

sub new {
    my $r_self = $_;[0];
    my $url = $_;[1];
    my $level = $_;[2];
    my $title = $_;[3];
    my $size = $_;[4];
    my $links_in = $_;[5]; # of internal bookmarks
    my $links_out = $_;[6];
    my $links_img = $_;[7];
    my $response_code = $_;[8]; # broken or suspended, timeout
    my $lks = $_;[9]; # array reference
    my $tokens = $_;[10]; # array reference
    my $response_time; #can compute the transfer rate
    my $content_type;
    my $last_modified=0;

    $r_self = { "URL" => $url,
                "LEVEL" => $level,
                "TITLE" => $title,
                "SIZE" => $size,
                "BOOKMARK" => $links_in,
                "LINKOUT" => $links_out,
                "LINKIMG" => $links_img,
                "LINKS" => $lks,
                "RS_CODE" => $response_code,
            };

    return $r_self;
}
```

"RS_TIME" => $response_time,
"LAST_MODIFIED" => $last_modified,
"TOKENS" => $tokens
};

bless ( $r_self, "JFILE");
return $r_self;

}

sub set_val {
  # input set type, value
  my ($obj, $attr, $vals) = @_;

  $obj->{attr} = $vals;
  # if ($attr eq "LINKS")
  # {
  #   $obj->{attr} = $vals;
  # }
  # else
  # {
  #   $obj->{attr} = $vals;
  # }

  #sort the links for better bin_search()
  sub sort_links {
    # my $debug = 1;
    my $obj = shift();
    my @temp = sort @{$obj->{get_val("LINKS")}};
    $obj->{set_val("LINKS", @temp)};
    if ($debug) {
      print "This is the sorted JFILE links \n\n";
      print join("\n", @temp);
      # <STDIN>;
    }
  }

  sub sort_token {
    my $obj = shift();
    my @result = ();
    my @temp = sort @{$obj->{get_val("TOKENS")}};

    my $tkn = pop(@temp);
    my $counter = 1;
    my $processed = 0;
    my $num_tks = scalar(@temp);
    while (defined($next_tkn = pop(@temp)) && $processed < $num_tks) {
      if ($next_tkn eq $tkn) {
        $counter++;
        $processed++;
      }
} else
{
    $tkn = $cntekr."\t\t\t"."$tkn."."""\n";  
    push(@result, $tkn);
    $tkn = $next_tkn;
    $cntekr = 1;
}

$obj -> set_val("TOKENS", \@result);
# print STDERR join("\n", @result);
}

sub is_linked {
    # my ($obj, $lk_url) = @;
    # my $miss1 = $lk_url."/";
    # my @lks = split(/\7, $obj->get_val('LINKS'));
    # foreach $lk (@lks)
    # {
    #     # my $miss2 = $lk."/";
    #     if ($lk eq $lk_url)
    #     || ($lk eq $miss1) || ($lk_url eq $miss2))
    #     {  
    #         return 1;
    #     }
    # }
    # return 0;
}

sub is_linked {
    #binary search an index from an array of string
    #input: a string, a sorted array
    #return $index or -1 for not found
    my $debug=1;
    my $obj = shift();  #FILE object
    my $str = shift();  #a URL
    my @sortedArray = @($obj->get_val("LINKS"));

    if ($debug) {print "bin search ......\n";}
    my $stop = scalar(@sortedArray)-1;  #top index
    my $btm = 0;  #bottom index
    my $mid = undef;  #mid index
    if ($debug) {print "my top = $stop; \n";}
    #<STDIN>;
    while($stop >= $btm) {
        $mid = int(($stop + $btm)/2);
        # if ($debug) {
        #     print ****BIN_SEARCH:\n\tmidIndex = $mid \n
        mid=$sortedArray[$mid]=";
        # print"\tttopIndex=$stop \t top=$sortedArray[$stop]\n";
        # print "\tbtmIndex=$btm \t btm=$bEventIDList[$btm]\n";
        # <STDIN>;
        # }
        if ([$sortedArray[$mid] eq $str] ||
            ($sortedArray[$stop] eq $str) ||
            ($sortedArray[$btm] eq $str)


A-2.2 JMAP.pm

## JMAP.pm

```perl
#!/usr/bin/perl

# JMAP.pm
require 'JFILE.pm';
require 5.004;
package JMAP;

=head1 COPYRIGHT

COPYRIGHT(R) Philip PENG, 1999-2004, peng_philip@hotmail.com
version 3.1.2

```

```perl

1;

```
$VERSION = "3.1.2";

=item 1.0.2 abstract method to fit all jmapping
=item 1.0.1 working version but too much repetition of code
=item 1.0.0 start constructing, building common method
=back

=cut

=head1 CLASS DESCRIPTION

*******************************************************************************
THIS MODULE provides basic functions for jmap processing

Attributes

%TYPES{\$type\} = $extensions
@URL_Queue

@Concept_Vews
$obj_files : save the object in an array, index is $seq
:Header_Symbols{$concept} = $symbols

@levels, contains # of url in the level

Methods

new() constructor
print_report()
sequence_jfile
get_dist_vals($attr, $datatype)
    jmap_header($concept, $symbol)
    jmap_level()
    jmap_url
    jmap_title
    jmap_size
    jmap_rs_code
    jmap_lk_cnt : links in, links out, img links
    jmap_dir_tree();

*******************************************************************************
=cut
use JFILE;

#global constant declaration
my $debug = 0;

sub new {
    # initialize the JMAP defaults
    my $r_jmap = $[0];
    my $r_jfiles = $[1];
    my $max_level = $[2];
    my $r_types = $[3];
    my $r_headers = $[4];
    my $r_links = $[5];  # sorted links set by order_url()
    my $log = $[6];  # log file name ; added for
    monitoring activities
    my $cnt_title = 0;
    # my $num_jfs = scalar @{$r_jfiles};
    # if (!defined($num_jfs)) { $num_jfs =0;};
    # "NUM_JFILE" => $num_jfs
    if ($log eq "")
    {
        $log = "jmap.log";
    }
    my @URL_CNT ;  # remember counters for each type of URL
    0 -- HTML, 1 -- OTHER, 2-- QUERY, 3-- IMG , 4 -- MAILTO, 5 -- FTP

    $r_jmap = {
        "JFILES" => $r_jfiles,
        "MAX_LEV" => $max_level,
        "TYPES" => $r_types,
        "HEADERS" => $r_headers,
        "URL_CNT" => 
            \@URL_CNT,
        "TITLE_CNT" => $cnt_title,
        "LINKS" => $r_links,
        "LOGFILE" => $log
    };

    bless ($r_jmap, 'JMAP');
    return $r_jmap;
}

sub limit_jfile {
    # in case of too many jfiles, we eliminate the outside links or links
    # without properties
    my $jmp = shift();
    my @jfiles = @{$jmp->get_val('JFILES')};
    my $rs_code;
    my @result=();

    foreach $jf (@jfiles)
    {
        $rs_code = $jf->get_val('RS_CODE');

        if (!defined($rs_code))
            
    }

}
```perl
sub order_url {
    # adjust jfile in certain order such as html, cgi, jpg, jepg, gif, ftp, mail, other
    my ($jmp, @urls) = @_;

    my $type = normal($jmp) -> get_val('TYPES');  # get types from jmp object
    my $url = '
    my @mailto = ();
    my @imgs = ();
    my @query = ();
    my @htmls = ();
    my @others = ();
    my @ftps = ();
    my @result = ();
    my $debug = 0;

    if ($debug)
    {
        print "\nOrdering urls: jfiles ", scalar(@jfiles);
        print "\nTypes: ", join("!", $type);
    }

    # classify url
    foreach $url (@urls)
    {
        if ( $url =~ m!$type['MAILTO']!!i )
        {
            push(@mailto, $url);
            if ($debug) { print "\n\nThis is a mail link.\n\n";}
        }
        elsif ( $url =~ m!$type['FTP']!!i )
        {
            push(@ftps, $url);
            if ($debug) { print "\n\nThis is a ftp link.\n\n";}
        }
        elsif ( $url =~ m!$type['IMG']!!i )
        {
            push(@imgs, $url);
            if ($debug) { print "\n\nThis is an image link.\n\n";}
        }
        # elsif ( $url =~ m!$type['QUERY']!! )  # fuck, I don't know why it does not work
    }
```


```
ellmessage = "<\&\#\+\]|\- asp|\ casc|\ exe|\ .pl|\ cqi-bin|\ .dll|"

} elsif ($url =~ m!(\|?\&\+\]|\- asp|\. asp|\. casc|\. exe|\. pl|\. cqi-bin|\. dll|) {  
push(@query, $url);  
if ($debug) { print "-->This is a query link.\n"; }  
} elsif ($url =~ m!^$type!(HTML)!i) {  
push(@htmls, $url);  
if ($debug) { print "-->This is a html link.\n"; }  
} else {  
push(@others, $url);  
if ($debug) { print "-->This is a other type link.\n"; }  
}

# if ($debug) { <STDIN>; }

} # end foreach

#make new url list and remember the counters  
my @COUNTER;  
push(@result, @htmls);  
$COUNTER[0] = scalar(@htmls);  
push(@result, @others);  
$COUNTER[1] = scalar(@others);  
push(@result, @query);  
$COUNTER[2] = scalar(@query);  
push(@result, @imgs);  
$COUNTER[3] = scalar(@imgs);  
push(@result, @mailto);  
$COUNTER[4] = scalar(@mailto);  
push(@result, @ftp);  
$COUNTER[5] = scalar(@ftp);  

# my @cnt = @{$jmp -&gt; get_val('URL_CNT')};  
# $cnt['HTML'] = scalar(@htmls);  
# $cnt['QUERY'] = scalar(@query);  
# $cnt['MAILTO'] = scalar(@mailto);  
# $cnt['OTHER'] = scalar(@others);  
# $cnt['FTP'] = scalar(@ftp);  
# $cnt['IMG'] = scalar(@imgs);  
# $jmp -&gt; set_val('URL_CNT', @cnt); # reset the values

# added for processing categorized urls sub sections  
$jmp -&gt; set_val("URL_CNT", @COUNTER);  
$jmp -&gt; set_val("LINKS", @result);  
return (@result);  
```

sub get_dist_vals {
    # process all distinct values from jfile objects
    # return a list with distinct values;
    # datatype must be "numeric" or "string", this is optional
    my $debug = 0;
    my ($obj, $view, $datatype) = @_; 
    my $temp = "";
    my $temp1="";
    my $temp2="";
    my @flist = ();
    my @result = ();

    if ($debug) {
        print STDERR "\n-->get_dist_val is processing distinct $view values.\n";
    }

    if (($view eq "URL") || ($view eq "TITLE") || ($view eq "TOKENS")) {
        $datatype = "string";
    } else {
        $datatype = "numeric";
    }

    my @jfs = @{$obj->get_val('JFILES')};
    my $num_jfs = scalar(@jfs);

    foreach $objf (@jfs) {  # get values
        $temp = $objf->get_val($view);
        if (defined($temp)) {
            # here is modified to accommodate token processing
            if ($view eq "TOKENS") {
                push(@result, @{$temp});  # token is an array reference
            } else {  # single values
                push(@result, $temp);  # push file's properties into arrays
            }
        }
    }

    if ($debug) {
        print STDERR "----> just looped all values.....\n";
        <STDIN>;
        print STDERR "----> start sorting the $view value.....\n";
    }

    # sorting arrays and eliminate duplicates
if ($datatype eq "numeric")
{
    @result = sort (@a=>@b)(@result);
    if ($debug) { print STDERR "\n------ finished sorting $view numbers ...\n"; }
}
elsif ($datatype eq 'string')
    # default to alphabetic order
    {
        @result = sort (@result);  # there will be problems for urls here?
        if ($debug) { print STDERR "\n======>>>>>finished sorting $view strings ...\n"; }
    }

    $temp1 = shift (@result);

    # there is an infinite loop here!!!!!!
    my @processed = 0;
    my $max_limit = scalar (@result);
    while (defined ($temp1)
        && $processed < $max_limit) #num_jfs)
        # process distinct list
        {
            $processed++;
            $temp2 = shift (@result);
            if ($debug)
            {
                print STDERR "---start making unique value ===\n";
                print STDERR "==== temp1: $temp1; temp2: $temp2; \n";
            }
            if (!@obj =~ is_equal ($datatype, $temp1, $temp2))
            {
                push (@flist, $temp1);
                $temp1 = $temp2;
            }
        }  # end while for both datatype

if ($view eq "URL")
{
    @flist = @obj -> order_url (@flist);
}
if ($debug)
{
    print STDERR "\n==== finished processing dist vals.\n";
    print STDERR "\nList is: ", join ("", @flist);
    <STDIN>;
}
if (($view eq "URL") || ($view eq "TITLE"))
{
    @obj -> set_val ("TITLE_CNT", scalar (@flist));
}
    return @flist;
}
sub set_val {
    my ($jmp, $attr, $val) = @_;  
    $jmp -> { $attr => $val; }
}

sub get_val {
    my ($obj, $concept) = @_;  
    return ($obj->{$concept});
}

sub jmap_header {  
    # type: concept view type, level: main view (0) or subtype, (1)  
    # num_dist: number of distinct values in the list  
    # return a row of string symbols
    my ($jmp, $concept, $num_hits_t, $num_dist, $num_hits_f ) = @_;  
    if ($debug) {  
        print "\nJmap header: obtained. $concept, $num_hits, $num_dist, \n";
    }  
    my $num_jf = scalar @{$jmp -> get_val ('JFILES')};
    if ($concept eq "0") { $symb_concept = "HTML";
    }  
    if ($concept eq "1") { $symb_concept = "OTHER";
    }  
    if ($concept eq "2") { $symb_concept = "QUERY";
    }  
    if ($concept eq "3") { $symb_concept = "IMG";
    }  
    if ($concept eq "4") { $symb_concept = "MAILTO";
    }  
    if ($concept eq "5") { $symb_concept = "FTP";
    }  
    my $symb = $($jmp -> get_val ('HEADERS')?>$symb_concept;  
    my $header ="";
    for ($i=0; $i < $num_jf; $i++) {  
        $header = $header."$symb\t";
    }  
    if ($concept eq "URL") {  
        $header .= 
        "$num_hits_f\t$num_hits_t\t$num_dist\t\"$concept.\"\t\"$symb_concept.\"\t\"$symb_concept.\"\n";
    }  
    elsif ($concept eq "0") # HTML  
    {  
        $header .= 
        "$num_hits_f\t$num_hits_t\t$num_dist\t\"HTML\"\t\"$symb_concept.\"\t\"$symb_concept.\"\n";
    }  
    elsif ($concept eq "1") # OTHER  
    {  
        $header .= "$num_hits_f\t$num_hits_t\t$num_dist\t\"OTHER TYPES\"\t\"$symb_concept.\"\t\"$symb_concept.\"\n";
    }  
    elsif ($concept eq "2") # QUERY  
    {  
        $header .= "$num_hits_f\t$num_hits_t\t$num_dist\t\"QUERY\"\t\"$symb_concept.\"\t\"$symb_concept.\"\n";
    }  
}
elsif ($concept eq "3") # IMAGE
{
    $header = "$num_hits_f\t$num_hits_t\t$num_dist_t\{IMAGE TYPES\}\t\{SIZE\}\n";
}
elsif ($concept eq "4") # MAILTO
{
    $header = "$num_hits_f\t$num_hits_t\t$num_dist_t\{MAILTO\}\t\{SIZE\}\n";
}
elsif ($concept eq "5") # FTP
{
    $header = "$num_hits_f\t$num_hits_t\t$num_dist_t\{FTP\}\t\{SIZE\}\n";
}
elsif ($concept eq "TOKENS")
{
    $header = "$num_hits_f\t$num_hits_t\t$num_dist_t\{TOKEN URL\}\t\{TOKEN STRING\}\t\{TITLE\}\n";
}
else
{
    $header = "$num_hits_t\t\t\t\t\{\$concept\}";
}
return $header;
}

sub is_equal {
    my ($Obj, $datatype, $val1, $val2) = @_;
    if ($datatype eq 'string')
    {
        return ($val1 eq $val2);
    }
    elsif ($datatype eq "numeric")
    {
        return ($val1 == $val2);
    }
    elsif ($datatype eq "url")
    {
        # print "Finding equal for url, $val1, $val2 \n";
        if ( ($val1 eq $val2 ) && ($val1 eq "$val1$"))
        {
            # print "return is 1\n";
            return 1;
        }
    }
    return 0;
}
sub jmap_url {
    #this function will map the url separately for
    # 0 HTML/TXT;
    # 1 UNKNOWN TYPES
    # 2 QUERY;
    # 3 GIF/JPEG;
# input: distinct sorted @URLS
# precondition: the URL must be sorted first using order_url()
my ($obj, $urls) = @_;  
my @cnt_url = @{$obj->[get_val("CNT_URL")]};
my $total_types = scalar(@cnt_url);

print STDERR "URL: total types are $total_types\n";

my $tmp_jmap_str = $obj->[jmapping("URL", $urls)];
my @splited_maps = split("
", $tmp_jmap_str);
my $jmap_str = pop(@splited_maps);  # the URL header

# add header for each subtype
for ($i=0; $i<$total_types; $i++)
{
    $jmap_str .= "$\n$boy->[jmap_header("$i")];
    for ($j = 0; $j < @cnt_url[$i]; $j++)
    {
        $jmap_str .= "$\n(pop(@splited_maps));
    }
}

$jmap_str .= "$\n"  # last line break
return $jmap_str;

sub jmapping_hz {  # mapping the digit lists in horizontal way
    my ($obj, $concept) = @_;  
    my @jfiles = @{$obj->[get_val("JFILES")]};
    my $num_jfile = scalar(@jfiles);
    my $num_hits = 0;
    my $jmap_str = "";
    my $val = 0;
    my $sum_val = 0;
    $obj->[logging("\n=====Start jmapping $concept ...", "new");
    $obj->[logging("\n=====There are total $num_jfile to be mapped,
please wait ...\n", "append");
    for ($i=0; $i<$num_jfile; $i++)
    {
        $obj->[logging("$i", "append");
        $val = @jfiles[$i]|[get_val($concept)];
        if (defined($val) && $val != 0)
        {
            $num_hits++;
            $sum_val += $val;
        }
        $jmap_str .= $val."\t";
    }
    if ($num_hits >=1){ $avg = $sum_val/$num_jfile;
    $jmap_str .= $num_hits."\t".$avg."\n";
my $header = $obj->[jmap_header($concept, $num_jfile, $num_hits)];
    $jmap_str = $header.$jmap_str;
    return $jmap_str;
sub jmap_url {
    my $jmp = shift();
    my @list_url = @_;
    my @jfiles = @{$jmp -> get_val("JFILES")};
    #
    my @list_url = @{$jmp -> get_dist_vals("URL")};
    my $num_url = scalar(@list_url);
    my $num_jfile = scalar(@jfiles);
    my $jmap_str = "";
    my $header = "";
    my $total_hits_f = 0;
    my $total_hits_t = 0;

    if ($debug)
    {
        print "\nJMAP_URL: = = = = = = = = =\n";
        print "JFILES: $num_jfile; Vals: $num_val \n";
        #
        <STDIN>;
    }

    my $url_cnt=0, $f_cnt=0, $t_cnt=0;
    my $jfs_cnt =0;
    my $curr_url, $f_url;

    for ($url_cnt = 0; $url_cnt < $num_val; $url_cnt++)
    {
        $curr_url = $list_url[$url_cnt];
        $f_cnt = 0;
        $t_cnt = 0;
        for ($jf_cnt =0; $jf_cnt < $num_jfile; $jf_cnt++)
        {
            $f_url = $jfiles[$jf_cnt] -> get_val("URL");
            if ($debug)
            {
                print "Get Links : ";
                print @{$jfiles[$jf_cnt] -> get_val("LINKS")};
                #
                <STDIN>;
            }

            $linked = $jfiles[$jf_cnt] -> is_linked($curr_url);

            if ($f_url eq $curr_url)
            {
                $jmap_str .= "f\t";
                $f_cnt++;
                if ($debug) {print "\nThis is a from page.";}
            }
            elsif ($linked ==1)
            {
                $jmap_str .= "t\t";
                $t_cnt++;
                #
                if ($debug)
                {
                    #
                    print "\nV V V V V I found linked.";
                }
<STDIN>;

else
{
    $jmap_str .= "\t";
    if ($debug)
    {
        print "\n xxxxxx nothing.";
    }
}
}

# inner for

$jmap_str .= "$t_cnt\t$t_cnt\t$t\curr_elem\n";
$total_hits_f += $f_cnt;
$total_hits_t += $t_cnt;

# outer for

$header = $jmp -> jmap_header("URL", $total_hits_f, $num_url);
$jmap_str = $header.$jmap_str;
return $jmap_str;

sub jmapping_token {
    my $obj = shift();
    my @jfiles = @{$obj->get_val("JFILES")};
    my $debug =0;
    if ($debug)
        print STDERR "\n*********************************MAPPING TOKENS *********************************\n";
    }

$obj -> logging("\n-----start mapping tokens ...
";
    my $total_jfile = scalar(@jfiles);

    my @tk_list = $obj -> get_dist_vals("TOKENS");
    my $jmap_str = ""

    my $prefix = "";
    my $l=0;
    my $num_found =0;
    my $total_tks =0;  # each row
    my $grand_total_tks =0;  # grand total
    my $total_dist_tks = scalar(@tk_list);
    if ($debug) { print STDERR "\n".
        print STDERR "\n***total distinct tokens:
        $total_dist_tks";
    }

    $obj -> logging("\n-----There are $total_dist_tks distinct token to be

    for ($cnt_token =0; $cnt_token < $total_dist_tks; $cnt_token++)
    {
        if ($debug) { print STDERR "\n
        $tk_list[$cnt_token]";
        # $obj -> logging("", "append")
        $obj -> logging("$cnt_token", "append")
    }
$total_tks = 0; # reset for each row
$suffix = ""; # reset prefix for each url-token pair
$num_found = 0;
for ($i=0; $i<total_jfile ; $i++) # inner for loop for one token line
{
    # $obj -> logging(">", "append");
    $num_found = $files[$i]-&gt;get_TokenFoundCnt($stk_list[$cnt_token]);
    if ($debug) { print STDERR "$n---Found # of Token: $num_found\n"; }
    $total_tks += $num_found;
    $grand_total_tks += $num_found;
    if ($num_found == 0)
    {
        $prefix .= "$t";
        if ($debug) { print STDERR "$n---Print a tab on the prefix.\n\n$prefix: $suffix\n"; }
    } elsif ($num_found == 1)
    { $prefix .= "t\t";
        if ($debug) { print STDERR "$n---Print a t on the prefix.\n\n$prefix: $suffix\n"; }
    } else
    { $prefix .= "$num_found"."t\t";
        if ($debug) { print STDERR "$n---Print a $num_found t on the prefix.\n\n$prefix: $suffix\n"; }
    } #end else

} # finish inner for for one token line
# end of one line, get the title
my $title = $obj-&gt;get_title($stk_list[$cnt_token]);
&lt;STDIN&gt;
$jmp_map_str = $prefix."l\t"."$total_tks."t".$stk_list[$cnt_token]."\t$title\n";
if ($debug) { print STDERR "$nThis token line: \n", $prefix."l\t"."$total_tks."t".$stk_list[$cnt_token]."\n"; }

my $counter = @$obj-&gt;get_val("URL_CNT");
$scounter[$j]= $total_dist_tks;
$obj-&gt;set_val("URL_CNT", $counter);
my $jmp_map_header = $obj-&gt;jmap_header("TOKENS",
    $grand_total_tks,
    $total_dist_tks,
    $total_jfile );
$jmp_map_str = $jmp_map_header.$jmp_map_str;
return $jmp_map_str;
}

sub jmapping {
# my $debug = 1;
# my ($obj, $view, $listval) = @_; # my $log = $obj->get_val("LOGFILE"); # for logging progress
# if ($open{LOG}, "$log")
# { #
# print STDERR "The log file cannot be opened\n";
# }

$obj->logging("\n===> processing jmap attribute $view .... \n", "append");
my @list_sval = (); # distinct value list
my $num_val = 0; # total value
my $num_related = 0; # related jfile numbers

if ($debug) { print "---> try to get distinct values ...\n"; }

if ($view eq "URL")
{
    @list_val = @{$listval};
    @list_val = $obj -> order_url(@list_val);
}
else
{
    @list_val = $obj -> get_dist_vals($view); # attribute list for
    jmapping
}

if (($debug) && ($view eq "URL"))
{
    print "---> jmap get list: ", join(" ", @list_val);
    
    <STDIN>;
}

my @jfiles = @{$obj -> get_val("JFILES")};
my $jff = $obj -> get_val("JFILES");

my $num_jfile = scalar(@jfiles);
my $jmap_str = "";
my $val ;
my $datatype = "";
my $curr_elem;
my $st_cnt = 0; # related item counter
my $f_cnt = 0;
my $total_hits_t = 0; # count t hits
my $total_hits_f = 0; # only count the f hits

if (($view eq "URL") || ($view eq "TITLE"))
{
    $datatype = "string";
}
else
{
    $datatype = "numeric";
}

if ($debug) { print "\nJmap datatype: $datatype, \n"; }
$num_val = scalar(@list_val);

if ($debug)
{
    print "\nList val $num_val: ", join("!", @list_val), "\n";
    <STDIN>;
}

my $i, $j, $title, $size;

for ($i = 0; $i < $num_val; $i++)
{
    $i_cnt =0;
    # reset for every row
    $t_cnt =0;
    $curr_elem = $list_val[$i];
    $title = "";
    $curr_elem_match = $curr_elem; # eliminate () when matching
    $size = $obj->find_size($curr_elem); # get the size for this page

    $curr_elem_match =~ s/\(/\{|/; # delimit {
    $curr_elem_match =~ s/\)/\}/; # }
    $curr_elem_match =~ s/\{|/\}/; # { }
    $curr_elem_match =~ s!\?!\?!!; #
    $curr_elem_match =~ s!\&!\&!!; # &
    $curr_elem_match =~ s!\*!\*!!; # *
    $curr_elem_match =~ s!\!\!\!\!\!\!\!; # **
    $curr_elem_match =~ s!\=\!\=!!; # ==
    $curr_elem_match =~ s!~\!\!\!\!\!\!\!; # !=

    if ($debug) { print "\n **********list val: $curr_elem, \n"; }
    for ($j = 0; $j < $num_jfile; $j++)
    {
        $obj_jf = $jff->[j];
        $val = $obj_jf -> get_val($view);

        if ($view eq "URL") # URL part is special since it is a
            start of relation
        {
            if ($debug)
            {print "\n\n\nt蓉ing to find linked url.";
            print "$s--selecting jfile val: $val, ";
            print "My linked pages are : ", join("\n", @{$obj_jf->get_val("LINKS")});
            print $obj_jf->get_val("LINKS");
            <STDIN>;
        }
    }

    my @links = @{$obj_jf->get_val("LINKS")};
if ($val eq $curr_elem)
{
    $jmap_str := "f\t";
    $title = $obj_if->get_val("TITLE"); # keep title
    $f_cnt++;
    if ($debug) { print "\nThis is a from page."; }
}

#
# elseif ($links =~ m!$curr_elem_match!) #
# elseif ($obj->is_linked($curr_elem, $links))
# elseif (($links =~ m!$curr_elem_match!)) ||
else  # $obj_if->is_linked($curr_elem)
{
    $jmap_str := "t\t";
    $t_cnt++;
    if ($debug) { print "\n\nI found linked.";
        <STDIN>;
    }
}

else
{
    $jmap_str := "t";
    if ($debug) { print "\nxxxxx nothing.";
    }
}

elseif ($obj->is_equal($datatype, $val, $curr_elem))
{
    $jmap_str := "t\t"; # mark relation
    $t_cnt++;
}
else
{
    $jmap_str := "t"; # escape
}

} # end for, inner loop

# Writing postfix to jmap
if ($view eq "URL")
{
    if ($title eq " n/a")
    {
        $title = "";
    }
    $jmap_str := "$f_cnt\tSt\tcnt\tt\tcurr\_elem\tt\title\tsize\n";
}
else
{
    $jmap_str := "$t_cnt\t\tt\tcurr\_elem\n";
}
$total\_hits\_f += $f_cnt;
$total\_hits\_t += $t_cnt;

} # end for
my $header = $obj -> jmap_header($view, $total_hits_t, $num_val, $total_hits_f);
$jmmap_str = $header -> $jmmap_str;
return $jmmap_str;
}

#sub is_linked {
# my ($obj, $curr lk, $links) = @_; 
# my @lks = split( //, $links);
# foreach $lk (@lks)
# {
# if ($lk eq $curr lk)
# {
# return 1;
# }
# }
# return 0;
#}

sub get_title {
# input a url, return a title or html file not obtained
my ($obj, $tkurl)= @_; 
my @Jfiles = @{$obj} -> get_val("JFILES") ;
my $num_files = scalar($Jfiles);
my @tkurl_split = split(\t/, $tkurl);
my $linkurl = $tkurl_split[0];
# print "The number of jfile is $num_files, get title for $linkurl\n";
my $title = "!this link was not processed.";

for ($i=0; $i<$num_files; $i++)
{
if ($obj -> is_equal("url", $linkurl, $Jfiles[$i] -> get_val("URL"))
{
# print "Found a title for this URL\n";
$title = $Jfiles[$i] -> get_val("TITLE");

last;
}
}
return $title;
}

sub find_size {
# input a url, return the size of this page
my ($obj, $tkurl)= @_; 
my @Jfiles = @{$obj} -> get_val("JFILES") ;
my $num_files = scalar($Jfiles);

# print "The number of jfile is $num_files, get title for $linkurl\n";
#<STDIN>;
my $size = 0;

for ($i=0; $i<$num_files; $i++)
{
if ($obj -> is_equal("url", $tkurl, $Jfiles[$i] -> get_val("URL"))
{
# print "Found a title for this URL\n";
$size = $Jfiles[$i] -> get_val("SIZE");
<STDIN>;
  last;
}
}
return $size;
}

sub logging {  # logging message to the window
  my ($obj, $msg, $opt) = @_;  
  my $log = $obj->get_val("LOGFILE"); # for logging progress
  #print STDERR "my log file is $log
"

  if ($opt eq "" || $opt eq "new")
  {
    open(Log, ">$log") || die "Cannot open log file\n";
  }
  else
  {
    open(Log, ">>$log");
  }
  print Log "$msg";
  close(Log);
}
1;

A-2.3  webjmapm.pm

#!/site/bin/perl
# PERL WIN32, JMAP BUILDER Main program for CGI
BEGIN { require 'JFILE.pm';
    require 'JMAP.pm';
    }

=head1 WEBSITE JMAP BUILDER 4

=head1 COPYRIGHT
#..........................................................................

COPYRIGHT:  Philip PENG, 1999-2005
JMAP is the trademark of Gen-Strategies, WMJ, 1995-2000

=head1 VERSION 4.1.0

VERSION HISTORY:
   2.0.3  infinite loop unsolved when only 1 url ,
   URL seems give 3-4 url with level 0
   order_url does not work in jmap.pm
   2.0.2  problems with jmap modules resolved, datatype
   2.0.1  dramatic abstraction was done, code is compressed
   2.0.0  draft version, no much abstraction for methods
   3.0.0  draft version, add token extraction and jmapping
   3.0.1  add vertical # attribute representation for link out,
     link img, bookmarks
   3.0.2  split URL columns to separate ones
3.1.0 add methods to do horizontal layout of number properties
3.1.1 add interface function to get stop message
4.0.0 Web cgi version
4.1.0 Revised Directory Structure and changed algorithms
4.1.1 Both GUI and webjmap4.pl can run on UNIX solaris and
WIN32
4.2.0 Add sendMail function to automate sending jMap

=cut

#main program starts
#jMap Configuration
#
my $SOS = "WIN32";
my $version = "Website jMap Builder 4.1.1";

#Email configuration to automate sending jMap to user
my $smtp_srv = "ticssmtp2.innovation.siemens.ca";
my $sender = "Philip.Peng@tic.siemens.ca";
my $receiver = "Philip.Peng@tic.siemens.ca";

#main program starts
#global Variable Declaration
#
my $JMAP_HOME = &getJmapHome();
print "JMAP home is $JMAP_HOME\n";
my $debug = 1;
my $config_file = "$JMAP_HOME/conf/jMap.conf";
my $start_time = time();
my $finish_time =0;
my $used_time =0;
my $log = "$JMAP_HOME/tmp/jmap.log";
my $root="";
my $max_level = 210;
my $curr_level =0;
my $max_url = 500;
my $jmap_style=1;
my $email_opt = 1;

my @URL_Q = ();
my @LEVEL = ();
my @TITLES = ();  #hold unique value of title
my @SIZES = ();
my @RS_CODES = ();
my @LINKS = ();  # an array to ensure every url we request will
be unique
my @LINKSIN=();
my @LINKSOUT=();
my @LINKSIMG =();
my @JFILES =();
my $HS_PARAM = 'j';
my $filename = "$JMAP_HOME/jmaps/jmap.txt";
my $stop = 0;  # user optional to stop after a while
my $cmdfile = "$JMAP_HOME/conf/stop.txt"; # gui can pass control to this program
my $jmap;
my %TYPE;
my %HEADER;
my $num_url_processed = 0;  # urls processed in each level
my $total = 0;  # total urls were processed
my $curr_level = 0;
my $size = 0;
my $title = "";
my $links_int = 0;  # bookmarks with #
my $links_out = 0;  # mailto also countered
my $links_img = 0;
my @TURLS = ();
my $htmlfile = "$JMAP_HOME/tmp/temp.html";

my $commandfile = "$JMAP_HOME/conf/commands.txt";
my @Parameters;
my @CMDS = ();
my $num_cmds = 0;
my $SSID = 0;  # session ID
my $num_sites = 0;  # number of joint map sites
my $report_file = "$JMAP_HOME/jmaps/JmapReport.txt";

############################################################
# Grand JMAP Variable Initialization
#
############################################################

my $grand_level = 0;
my @GRAND_JFILES = ();
my @GRAND_LINKS = ();

############################################################
# processing view types
$TYPES = ( "HTML" => "<html hommes = "text/html" http://$+
"HTML" => "<html homme = "text/html" http://$+
"IMG" => "</image > " gif",
"QUERY" => "[?\&\+]\.asp\|\.as\|\.cgi\|\.cgi-
bin\|\.exe\|\.dll\|\.pl",
"FTP" => "ftp:///",
"MAILTO" => "mailto:",
"OTHER" => "(\$+)",
"SKIP" => "\.doc\|\.ps\|\.pdf\|\.ppt\|\.xls"
);  # current supported types

# view header symbols
$HEADER = (
  "URL" => "L",

97
"LEVEL" => "A",
"level" => "v",
"ADDRESS" => "M",
"HTML" => "v",
"OTHER" => "v",
"IMG" => "v",
"QUERY" => "v",
"MAILTO" => "v",
"FTP" => "v",
"SIZE" => "N",
"LINKS" => 'N',
"BOOKMARK" => "v",
"LINKOUT" => "v",
"LINKIMG" => "v",
"TITLE" => "N",
"RS_CODE" => 'N',
"TOKENS" => "T"
);

########################################################################
#
# Variable Declaration
#
########################################################################

########################################################################
#
# Main function starts here
#
########################################################################

&set_stop(0); # overwrite the previous stop file
&read_config(); # read in configuration from jMap.conf
&read_cmds($commandfile); # read in process request
print STDERR "$num_cmds
Number of commands is $num_cmds
";
open(REPORT, ">$report_file");
# print REPORT "Session ID:
URL:	JMAP Filename"
for(my $cmd_counter=0; $cmd_counter<$num_cmds ; $cmd_counter++)
{
    my @cmds = split("/	/", $CMDS[$cmd_counter]);
    $SSID = $cmds[0]; # session ID
    $Parameters[0] = $cmds[1]; # URL
    $Parameters[1] = $cmds[2]; # max_level
    $Parameters[2] = $cmds[3]; # max_url
    $Parameters[3] = $cmds[4]; # jmap_style
    $Parameters[4] = $cmds[5]; # Associate sites jointed maps indicator
    $Parameters[5] = $cmds[6]; # email notification option
    print STDOUT "SID = $SSID, URL = $cmds[1]; \n"
    # set_session_filenames($SSID); #$sid = sid
    $filename = "$JMAP_HOME/jmaps/jmap."."SSID".".txt";
    if ($debug) { print STDOUT "Jmap file name is $filename\n";
        if ($SOS eq "WIN32"){
            print REPORT "$SSID\t$cmds[1]\t", &get_dos_filename($filename), "\n";
        } else { #OS X
            print STDOUT "$SSID\t$cmds[1]\t", &get_unix_filename($filename), "\n";
        }
    } else {
        # print STDOUT "$SSID\t$cmds[1]\t", $filename, "\n";
    }
else {
    print REPORT "$SSID\t$cmds[1]\t", $filename, "\n";
}
print STDERR
"\n******************************************************************************\n"
    print STDERR "\n\n>>>Start process website $cmd_counter: $cmds[1] ..\n\n"
# &process_url(@Parameters); # retrieve URL, token and save them to
JFILES
    if ($Parameters[4] == 1 )
    {
        $num_sites++;
        print STDERR "Saving Parameters for cross mapping\n";
# <STDIN>;
        &save_current_params(); # save current parameters for joint site
    
processing
    }
    if ($Parameters[5] == 1 )
    {
        $email_opt =1;
    }
# print STDERR "JMAP file name is $filename\n";

    print STDERR "\n\n>>>Start process JMAP for site $cmds[1] ..\n"
# &process_jmap(); # process jmap using current global variables
}
if ($num_sites >= 1 )
{
    $filename = "$JMAP_HOME/jmaps/jmap."."cSSID".".txt";
    print STDERR
"\n******************************************************************************\n"
    print STDERR "\nStart cross-mapping between the websites ...
";
    &show_time("\nStart cross-mapping between the websites ...
"");
    $curr_level = $grand_level;
@JFILES = @GRAND_JFILES;
@LINKS = @GRAND_LINKS;
# print STDERR "Total Jfile = ", scalar(@GRAND_JFILES), "\n";
# print STDERR "Total Links = ", scalar(@GRAND_LINKS), "\n";
    &process_jmap();
    print REPORT "$SSID\t$\nCross mapping $num_sites sites\t",
    &get_dos_filename($filename), "\n";
}
close(REPORT);

# print last message before close
open(LOG, ">$log") or die "Cannot open log file to write\n";
    print LOG "\n\nJMAP 4.1.0 has finished jmapping $num_sites 
websites.\n"
    print LOG "Total used time is $used_time seconds. \n";
    print LOG "Average used processing time per url is $used_time /
$total second.\n"
    print LOG "Thank you using JMAP to JMAPing website.\n"
    print LOG "Please click View report to see the report.\n"
close(LOG);
unlink($htmlfile);
print STDERR "\nJMAP completed mapping the websites ...\n\n";
99
&set_stop(0);
0;

#main program ends

# Subroutine implementation starts here
#
# getJmapHome
#
sub getJmapHome{
    use Cwd;
    chdir("..");
    my $home = getcwd();
    chdir("bin");
    return $home;
}

sub read_config {
    # read in jMap.conf configuration parameters
    open (CONF, "<$config_file") ||
        print STDERR "Reading configuration file $config_file file failed.
"

    # SOS = "WIN32";  # WIN32 or UNIX
    # $version = "Website jMap Builder 4.1.1";
    while (<CONF>) {
        chomp();
        if ( /OS\=\s+UNIX/) { SOS = "UNIX"; }
        elsif ( /OS\=\s+WIN32/) { SOS = "WIN32"; }
        elsif (/Outgoing SMTP Server\=\s+([-\w.\)]+)/) {
            $smtp_server = $1
        } elsif (/Reply Email address\=\s+([-\w.\)]+)/) {
            $sender = $1;
        }
    }
    close(CONF);
    print STDERR "OS: $SOS
";
    print STDERR "SMTP SERVER: $smtp_server
";
    print STDERR "Reply Email: $sender
";
    # Email configuration to automate sending jMap to user
}

sub read_cmds {
    # read customer commands
    my $cmd_file = shift();
    open(CMD, "<$cmd_file");
    while (<CMD>) {
        chomp();  # get ride of new line characters
        if ($_ =~ /\d+\t$/) {
            $CMDS[$num_cmds++] = $_
        } elsif (/mailto:\s+([-\w.\)]+)/) {
            print STDERR "Error: Unexpected mailto command.
";
        } else {
            print STDERR "Error: Unknown command: \$_
";
        }
    }
}
$receiver = $l;  # email receiver
  print STDERR "Email to: $receiver\n";
}
}
close(CMD);
print join("\n", @CMDS);
}

sub set_session_filenames {
  # set up temp file name, output file name
  my $sid = shift();
  $filename = "jmap"."$sid".".txt";
  # print STDERR "New file name is $filename\n";
}

sub save_current_params {
  # save current session for future joint mapping

  if ($curr_level > $grand_level)
    { $grand_level = $curr_level; }
  push(@GRAND_LINKS, @LINKS);
  print "Total saved LINKS = ", scalar(@LINKS), " Grand ",
    scalar(GRAND_LINKS), "\n";
  push(@GRAND_JFILES, @JFILES);
}

sub process_url {
  my ($rt, $smx_level, $smx_url, $sjmap_sty) = @_;  
  # $root = $rt;
  $smx_level = $smx_level;
  $smx_url = $smx_url;
  $sjmap_style = $sjmap_sty;
  # $email_opt = $email_chk;

  print STDERR "$root, $smx_level, $smx_url, $sjmap_style, $email_opt \n";
  use LWP::UserAgent;
  use LWP::Simple;
  use HTML::Parser;
  use HTML::HeadParser;
  use HTML::LinkExtor;
  use URI::URL;
  use JFILE;
  use JMAP;

  # my $start_time = time();
  # my $finish_time = 0;
  # my $used_time = 0;
  # my $sjmap_style = 0;
  # my $log = "$JMAP_HOME/tmp/jmap.log";
  # my $root = $args[0];  # user input url as root of the server
  # my $max_level = $args[1];  # user input for the maximum level

  my $excel_limit = 230;  # maximum 230 columns
if (!defined($max_level))
{
    $max_level = 10;
    if ($debug) {print "Default level to 4.\n";}
}

if (!defined($max_url))
{
    $max_url = $excel_limit;  # this is the max column for Excel
97
    if ($debug) {print "Default number of file to $max_url.\n";}
}

# print STDERR "Default jmap style is $jmap_style.\n";
if (!defined($jmap_style))
{
    $jmap_style = 1;  # this is the max column for Excel 97
    if ($debug) {print STDERR "Default jmap style is $jmap_style.\n";}
}

open(LOG, ">$log") || die ("cannot open log file\n")
print LOG "Start processing web site urls....\n"
close(LOG);

# Reset global variables for this current session
@URL_Q = ();
@LEVEL = ();
@TITLES =();  # hold unique value of title
@SIZES = ();
@RS_CODES = ();
@LINKS = ();  # an array to ensure every url we request will be
unique
@LINKSIN=();
@LINKSOUT=();
@LINKSIMG =();
@JFILES =();
$jmap =

$num_url_processed = 0;  # urls processed in each level
$total=0;  # total urls were processed
$curr_level = 0;
$size = 0;
$title = "";
$links_int = 0;  # bookmarks with #
$links_out = 0;  # mailto also countered
$links_img = 0;
@TURLS=();
$htmlfile = "$JMAP_HOME/tmp/temp.html";

$url = $root;

# tailor $root for filter string
if ($root =~ /\(^http:\/\/\/\S+\\/\)\S+(\$TYPE{'HTML'}|\$TYPE{'QUERY'})\S/i)
{

102
$root = "$1; 
}
else if ($root =~ /^(http://\S+[^/\/]$)/)
{
    $root = "$1"/"; 
}

print STDERR "The root is selected as $root.\n";
push(@LINKS, $root);
push(@URL_Q, $root);
# change root to a closest domain name
$root = get_domain_name($root);

print STDERR "The domain filter is $root\n";
$LEVEL[0]=1;

$ua = new LWP::UserAgent;
# header parser for HTML
$u = HTTP::Headers->new();

# ($curr_level <= $max_level) &&
while ($total < $max_url)
{
    &check_stop();    # check stop flag
    if ($stop) {
        &show_time("Website jMap Builder is stopping.", 1);
        last;
    }
    $myurl=shift(@URL_Q);
    if (!defined($myurl))
    {
        if ($debug) print "\nAll links have been processed.\n";
        last;
    }  # if no more links, out of loop

    # session initialization
    undef $response_code;
    undef $protocol;
    $size = 0;
    $links_out = 0;
    $links_int = 0;
    $links_img = 0;
    $title = "";
    undef $rs_time;
    $total++;
push(@URLS, $myurl);

    my $msg = "\nProcessing URL Number: $total \nPage url : $myurl ....\n";
    $msg .= "\n=====Starting extracting links from page.\n";
    print STDERR "$\nNow processing ".$total."th url : $myurl ....\n";
    &show_time($msg, 1);

    @new_links =();    # hold all A type links
    $num_url_processed++;
if ($num_url_processed > $LEVEL($curr_level))
{
    $curr_level++;
    $num_url_processed = 1;  # reset for current level
    if ($curr_level > $max_level) { last; }
}

if ($debug)
{
    print "\n>>>>>>>Current level is $curr_level and ",
          $num_url_processed, " is been processing.\n";
}

# extract new links from only HTML, QUERY type links

# Make the parser.Unfortunately, we don't know the base yet (it might be
different from $url)
$hp = HTML::LinkExtor->new(&cb_save_links);

# header parser for HTML
$hp = HTML::HeadParser->new($hp);

# Request document and parse it as it arrives
$req = HTTP::Request->new(GET => $myurl);
$response = $ua->request($req, 
&cb_request);

$response_code = $response->code();
$title = $hp->header('Title');
if (!defined($title))
{
    $title = " n/a";
}
undef $hp;
undef $sh;
undef $hp;

if ($response -> is_success)
{
    # Expand all image URLs to absolute ones
    my $base = $response->base;
    @new_links = map { "$_" = url($_, $base)"->abs; } @new_links;
    ($links_out, $links_int, $links_img) = &links_out($myurl,
    @new_links);
    &enqueue(@new_links);  #push links to @URL_Q, maintain
    @LINKS
}  # end of if success response

# ------------------------------------------------------------------------
#
# save JFILE
# ------------------------------------------------------------------------
#
#$lkk_str = join(" ", @new_links);
print "Starting map tokens in this page....\n";
my @jf_tokens = &map_token($myurl);
$sfile = new JFILE( $myurl,
$curr_level,
$title,
$size,
$links_int,
$links_out,
$links_img,
$response_code,
$lk_str,
\@new_links,
  \@Jf_tokens
);
if (scalar(@new_links) >= 1) {
    $jfile->sort_links();  # sorting links for binary search
} push(@JFILES, $jfile);  # first file obj
}

sub process_jmap {
    # global @JFILES, $curr_level, \$TYPES, \$HEADER
    # make JMAP object to process jmap
    if ($debug) { print "\nTotal Jfiles \n\n\n"; }
    open(LOG, ">$log")
    print LOG "\n\nStart jmapping this website, please be patient...\n"
    close(LOG);
    my @full_url = ();
    \$jmap = JMAP->new(@JFILES,
        $curr_level,
        \$TYPES,
        \$HEADER
    );
    @full_url = \$jmap->order_url(@LINKS);
    \$jmap->set_val("LOGFILE", $log);
    \#process first line parameters
    \$HS_PARAM = \$jmap->get_val("URL_CNT");
    my $html_cnt = $HS_PARAM[0];
    my $other_cnt = $HS_PARAM[1];
    my $query_cnt = $HS_PARAM[2];
    my $imgs_cnt = $HS_PARAM[3];
    my $mailto_cnt = $HS_PARAM[4];
    my $sftp_cnt = $HS_PARAM[5];
    my $token_cnt = $HS_PARAM[6];

    my \$jmap_str = "";
    \$jmap_str .= \$jmap->jmapping("LEVEL");
    \$jmap_url = \$jmap->jmapping("URL", \@full_url);  # @LINKS can be used
    \$jmap_str .= \$split_url($jmap_url);
    \$jmap_str .= \$jmap->jmapping("TITLE");
    if ($jmap_style == 0) {
        \$jmap_str .= \$jmap->jmapping("BOOKMARK");
        \$jmap_str .= \$jmap->jmapping("LINKOUT");
    }
}
null
# print LOG "Total used time is $used_time seconds. \n";
# print LOG "Average used processing time per url is ",
$used_time/$total, " second.\n";
# print LOG "Thank you using JMAP to JMAPing website.\n";
# print LOG "Please click View report to see the report.\n";
# close(LOG);
# unlink($htmlfile);
} # end of process_jmap

sub jmap_sites {
  # jmap cross all processed sites
}

#########################################################################
# #
# Get options from user
#
##########################################################################
sub check_stop {
  # read user message to stop the program
  open(STOP, "<$cmdfile") || return;
  my $cmd = <STOP>; # currently only one line
  close(STOP);

  if ($cmd =~ /STOP\=1/) {
    $stop =1;
    unlink($cmdfile);
    my $msg = "\n\n======> Now, JMAP 4.0 is trying to stop after the
last request....\n";
    &show_time($msg, 0);
  } else {
    $stop = 0;
  }

  return;
}

sub set_stop {
  #input 0 or 1
  my $stp = shift();
  open(STOP, ">$cmdfile") || return;
  print STOP "STOP=$stp\n"; # currently only one line
  close(STOP);
}

sub cb_request {
  my ($str, $res, $prot) = @_;  
  $p->parse($_[0]);
  $hp->parse($_[0]);
  $size += scalar(split(/, $_[0]));
sub cb_save_links {
    # push found links to two different categories
    my($tag, %attr) = @_;  
    push(@new_links, values %attr); 
    return;
}

sub links_out {
    my ($curr_url, @lks) = @_; 
    my $int_counter = 0;    # internal bookmark
    my $ext_counter = 0;    # it also includes itself
    my $img_counter = 0;
    my @pieces = ();
    my @1k url="";
    foreach $1k (@lks) {
        if ($1k =~ /\#\S/i) {
            @pieces = split(\#//, $1k);
            $1k_url = shift(@pieces);
            if ($1k_url eq $curr_url) {
                $int_counter++;
            } else {
                $ext_counter++;
            }
        } elsif ($1k =~ m!TYPES\{!s/!i) {
            $img_counter++;
        } else if ($1k =~ m!TYPES\{'HTML'|QUERY'}!i) {
            $ext_counter++;
        }
    return ($ext_counter, $int_counter, $img_counter);
}

sub is_duplicate {
    # input a element and an array, return true if found in the current list
    else false
    my ($c_url, @MYLIST) = @_; 
    my $miss1=$c_url="/";        # some missing /
    my $miss2;
    foreach $1_url (@MYLIST) {
        $miss2 = $1_url="/";
        if ( $1_url eq $c_url || $miss2 eq $c_url || $miss1 eq $1_url ) {
            return 1;
        }
}


}
return 0;
}

sub enqueue {
# input extracted html links
# filter out internal bookmarks, mailto, ftp, etc links
# make sure the value is unique in the @LINKS
my @new_lks = @_;
my @LK=(); # select links
my @LK_OUT=(); # non-selected links
my @lks_new = ();
foreach $lk (@new_lks) {
    if ($lk =~ /^<\S+.*<\S+$/)
        $lk = $1; # strip internal bookmarks
    push(@lks_new, $lk); # rearrange chopped links
    if (!&is_duplicate($lk, @LINKS)) {
        push(@LINKS, $lk);
    }
    # ftp links are dangerous big sometimes, get rid of ftp mailto, some
    # unknown types
    if ($lk =~ /<root>/) # separate current site from others
        push(@LK, $lk);
    } # inner if
    else  # get outside links to make a jfile
        push(@LK_OUT, $lk);
}
# end outer if

@new_lks = @lks_new; # replace the links with new stuff
push(@URL_Q, @LK);
# increase # of url in queue for next level
$LEVEL[$curr_level+1] += scalar(@LK);
}

sub map_token {
    use LWP::UserAgent;
    use HTML::LinkExtor;
    use URI::URL;
    use HTML::TokeParser;

    # input URL as the first argument
    my $jurl = shift();
    my $html;
my $MSG = "\n======>Start extracting tokens from page: $jfurl\n";
&show_time($MSG, 0);
my @Tokens=();

    print STDERR "Get token for : $jfurl \n";
    $html= get($jfurl);
#    $shtmlfile= "temp.html";   # already defined in main
    open(HTMLFH, ">$shtmlfile");
    print HTMLFH "$html\n";
    close(HTMLFH);

    $ua = new LWP::UserAgent;
    my $links = $ua->request(HTTP::Request->new(GET => $jfurl));
    # Expand all image URLs to absolute ones
    my $base = $links->base;

    $p = HTML::TokeParser->new($shtmlfile) || die "Cannot open html page\n";
while (my $token = $p->get_tag("a"))
    {
        my $url = $token->[1]"href";
        $url = url($url, $base)->abs;
        my $text = $p->get_trimmed_text("/a");
        my $tokn = $url."\t$text";
        push(@Tokens, $tokn);
    }

    $MSG = "\n======>Finished token extraction from this page.\n...\n";
    &show_time($MSG, 0);
    return @Tokens;
}

sub find_symbol
{
    my $urlin = shift();
    for ($i=0; $i< scalar(@full_url); $i++)
    {
        if ($urlin eq $full_url[$i])
            return "$s".$i;
    }
}

sub show_time{
    my $msg = shift();
    my $is_new = shift(); # 1 for new message, otherwise append only
    $finish_time = time();
    $used_time = $finish_time - $start_time;
    if ($is_new == 1)
    {
        open(LOG, ">$log");
        print LOG "Processing session\t", $SSID;
print LOG "Starting elapsed time: ", $used_time, " seconds."

print LOG $msg;
} else {
    open(LOG, ">$log");
    print LOG "Current elapsed time: ", $used_time, " seconds.\n"
    print LOG $msg;
}

close(LOG);

sub split_url {
    #this function will map the url separately for
    # 0 HTML/TXT;
    # 1 UNKNOWN TYPES
    # 2 QUERY;
    # 3 GIF/JPEG;
    # 4 MAILTO;
    # 5 FTP;

    # input: $url_map, and $links
    # precondition: the URL must be sorted first using order_url()
    my $url_map = shift();

    my @total_types = scalar(@SHS_PARAM);

    my @splited_maps = split(/\n/, $url_map);

    my $tmp_jmap_str = shift(@splited_maps)."\n"; # the URL header

    #add header for each subtype
    for (Sj = 0; Sj < @total_types; Sj++) {
        $tmp_jmap_str .= $jmap->jmap_header($j, 0, SHS_PARAM[$j], scalar(@FILES));
        for (Sj = 0; Sj < SHS_PARAM[$j]; Sj++) {
            $tmp_jmap_str .= shift(@splited_maps)."\n";
        }
    }

    return $tmp_jmap_str;
}

sub get_domain_name {
    my $dn = shift();
    #print "this is the input root ", $dn, "\n";
    my $lastIndex = index($dn, "/");
    #print "The reverse index for last / is ", $lastIndex, "\n";
if ($lastIndex > 11){
  $dn = substr($dn, 0, $lastIndex);
}

# print "this is the output root ", $dn, "\n";
return $dn;
}

sub get_dos_filename {
  my $filename = shift();
  my @dos_filenms = split(/\//, $filename);
  return join("\\", @dos_filenms);
}

sub send_email {
  # send email to customer
  use MIME::QuotedPrint;
  use MIME::Base64;
  use Mail::Sendmail 0.75; # doesn't work with v. 0.74!
  if ($receiver eq "") {
    return;
  }
  print STDERR "\n\nSending jMap to the user.....\n"
  my $jmapfile = shift(); # This is the perl executable
  my $hostname = 'hostname';
  my $subject = "jMap for session $SSID, $root";
  my $Thankyou = "$version mailer: \n";
  $Thankyou .= "\njMap Session: $SSID\n";
  $Thankyou .= "\njMap Root URL: $root\n";
  $Thankyou .= "\njMap file attached: $jmapfile\n";
  $Thankyou .= "\nThank you for using $version.\n";
  $Thankyou .= "\nPlease see attachments for the jMap.\n\n";
  $Thankyou .= "\nFrom you jMap host @$hostname\n";

  &show_time("Sending email to $receiver .....\n$SSID\n$jmapfile\n", 1);
  my $mail = {
    SMTP => $smtp_svr,
    from => $sender,
    to => $receiver,
    subject => $subject,
  };

  $boundary = "===" . time() . "===";
  $mail{'content-type'} = "multipart/mixed; boundary="$boundary\"";

  $message = encode_qp( "$Thankyou" );

  open (F, $jmapfile) or die "Cannot read $jmapfile: $!";
  binmode F; undef $/;
  $mail{body} = encode_base64(<F>);
  close F;

  $boundary = '---'.$boundary;
$mail{body} = <<END_OF_BODY;
$boundary
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

$message
$boundary--
END_OF_BODY

sendmail($mail) || print "Error: $Mail::Sendmail::error\n";

}
1;

A-3 Web JMAP Builder Mail Server

A-3.1 pop3svr.pl

#!/usr/bin/perl

# How to obtain jmap email Request with Net::POP3
# Copyright 2000, Philip Peng, W. M. Jawarski,
# Email: WMJ@gen-strategies.com
# Created 02/2/02
# This script is designed to check pop3 jmap request
# email and call webjmap to process request and send
# jmap result back to user.
# Require module: Net::POP3,
# Windows: IMAP

use strict;

print "Content-type: text/plain", "\n\n";

use Net::POP3;

# This debug flag will print debugging code to your browser, depending
# on its value
# Set this to 1 to send debug code to your browser. Set it to 0 to turn
# it off.
mym $DEBUG = 0;

if($DEBUG)
{

$i = 1;
opendir(STDERR, "&STDOUT");
"

# Set this variable to your POP3 server name
my $ServerName = "mail.mondenet.com";
# Initiate the mail transaction
my $Username = "ppeng";
my $Password = "bmcy4ax7";
my @Q_CMDS = (); # a queue for all incoming commands
my $RequestKey = "Website jMap Building Request";

my $jmap_Path = &getJmapHome();
my $session_file = "$jmap_Path/conf/"."session.txt";
my $config_file = "$jmap_Path/conf/jMap.conf";
my $command_file = "$jmap_Path/conf/commands.txt";
my $stop_file = "$jmap_Path/conf/stop.txt";
my $pop3srv_logfile = "$jmap_Path/log/pop3srv.log";
my $log_msg = "Website jMap Building Mail Server log.".\"\n\n";
my $webjmapm_exe = "webjmapm.pl";

if (!(-e $pop3srv_logfile)) {
    if (open(POPLOG, ">$pop3srv_logfile"){
        print POPLOG $log_msg;
    }#end if
}#end if

my $interval = 3; # every 15 minutes to check email
my $SSID = 0;

# read in above configuration file path and vars to update
read_config();
set_stop("0"); # set stop flag to 0

while (1) {
    # flush command Queue
    sleep($interval);
    print get_timestamp(), ": POP3 server is awake now....\n";
    write_log("POP3 server is awake now....");
    @Q_CMDS = ();

    if (check_stop()) { # jMap is busy
        print "jMap Pop3 Server is stopping now ...\n";
        write_log("jMap Pop3 Server is stopping now ...");
        exit(0);
    }

    # Create a new POP3 object
    write_log("Try to connect to server.");
    my $pop3 = Net::POP3->new($ServerName, Debug => $DEBUG);

    # If you can't connect, don't proceed with the rest of the script
    # die "Couldn't log on to server.\n" unless $pop3;
    if (!$defined($pop3)) {
        print "--> Couldn't log on to server.\n" ;
write_log("Couldn't log on to server.");
next;
}

my $Num_Messages = $pop3->login($UserName, $Password);
print " ---> Login to POP3 server is successful. Total $Num_Messages
mails.\n";
write_log("Login to POP3 server is successful. Total $Num_Messages
mails.");
# Get the list of messages
my $Messages;
my $msg_id;
SSID = &get_sid();
my $init_sid = $SSID;
$Messages = $pop3->list();

# Parse each message header for "From" and "Subject" fields
foreach $msg_id (keys($Messages))
{
    my $MsgContent = $pop3->top($msg_id, 20);
    # print join("\n", @{$MsgContent});
    # <STDIN>;
    if (extract_cmds($MsgContent)) {
        $pop3->delete($msg_id);
    } else {
        $SSID = $init_sid; # roll back sid
    }
} #end foreach

# Close the connection
$pop3->quit();
if ($SSID > $init_sid) {
    &save_sid($SSID);
}

my $num_cmd = scalar(@Q_CMDS);
my $processed_cmds=0;
if ($num_cmd >0) { # process all the commands from Queue
    while (write_command() || $processed_cmds < $num_cmd)
    {
        system("perl $webjmap_exe");
        $processed_cmds++;
        write_log("Webjmapm.pl was called, the $processed_cmds
request was processed.");
    }
} #end if
} #end while

print "Server exits, no more service is available.
\n\n";
write_log("Server exits, no more service is available.\n\n\n");
exit(0);
1;

# This subroutine parses the "From" and "Subject" fields of a message header
sub extract_cmds
{  
  # Assign parameter to a local variable  
  my ($lines) = @_;  
  
  # Declare local variables  
  my ($from, $line, $subject, $return_addr);  
  my @CMDS = (); #my local array to store commands  
  my $cmd = "";  
  # Check each line in the header  
  foreach $line (@lines) {  
    if ($line =~ m/^From: (.*)$/) {  
      # We found the "From" field, so let's get what we need  
      $from = $1;  
      if ($from =~ m/\<(.*)\>/) {  
        #  
        print "Email: $1\n";  
        $from = $1;  
        push(@CMDS, "mailto:$t$from");  
        # $cmd = "mailto:$t$from\n";  
      }  
      $from =~ s/\|<.*>/g;  
      $from = substr($from, 0, 39);  
    } else if ($line =~ m/^Subject: (.*)$/) {  
      # We found the "Subject" field, so let's get what we need  
      $subject = $1;  
      # $subject = substr($subject, 0, 29);  
    } else if ($line =~ m/(http://\/\/)\(.*\)\(.*\)\(.*\)\(.*\)\(.*\)\(.*\)\(.*\)\(.*\)\(.*\)\(.*\)/) {  
      $SSID++;  
      push(@CMDS, "$SSID$t$1"."$2$t"."$3$t"."$4$t"."$5$t"."$6$t");  
      # $cmd .= "$SSID$t$1"."$2$t"."$3$t"."$4$t"."$5$t"."$6$t";  
    }  
    # If we have parsed the "From" and "Subject" field, then we don't  
    # need to keep on going. Let's quit the loop here.  
    # list if( defined($subject) && defined($from));  
  }  
  # Print the result  
  printf "\n\nFrom: "-40s \nSubject: "-s\n", $from, $subject;  
  
  if ($subject =~ m/$RequestKey/i) {  
    #print commands  
    my $cm = join("\n", @CMDS);  
    print "Yes, this is a Request. \n$cm \n";  
    push(@Q_CMDS, $cm);  
    return 1;  
  }  
  return 0;  
}  
# end: PrintList()  

sub getJmapHome{  
  use Cwd;

116
```perl
chdir("..");
my $home = getcwd();
chdir("bin");
return $home;
}

sub get_sid {
    #$session_file = "session.txt";
    open(SESS, "<$session_file") || die("Cannot open session file for validation.
")
    my $sess_id=0;
    while (<SESS>){
        chomp();
        if (/^\d+$/) {
            $sess_id = $1;
            last;
        } # end else
    } #end while
    close(SESS);
    return $sess_id;
}

sub save_sid {
    my $sid3 = shift(@_);
    $sid3 ++ = 1;
    open(SESSL, ">$session_file") || die("unable to open session.txt to overwrite
");
    print "This is the new sid: ", $sid3;
    print SESSL "$sid3
";
    close(SESSL);
}

sub write_command {
    my $mresult = 0;
    open(CMD, ">$command_file") || return 0; # failed, so give up
    my $cmdss = shift($@Q_CMDS);
    print "****Writing commands to $command_file
$cmdss
";
    if (defined($cmdss)) {
        print "Writing commands to $command_file
$cmdss
";
        print CMD $cmdss;
        $mresult = 1;
    } else {
        print "No more commands to $command_file
$cmdss
";
        $mresult = 0;
    }
    close(CMD);
    write_log("This is the request:
$mresult");
    return $mresult;
}

sub set_stop {
    #input 0 or 1
    my $stp = shift();
    open(STOP, ">$stop_file") || return;
    print STOP "STOP=$stp
"; # currently only one line
    close(STOP);
}
sub check_stop {
    # read user message to stop the program
    open (STOP, "<$stop_file") || return;
    my $cmd = <STOP>; # currently only one line
    close(STOP);

    if ($cmd =~ /STOP=*1/) { return 1; }
    else { return 0; }
}

sub read_config {
    # read in jMap configuration parameters
    open (CONF, "<$config_file") ||
        print STDERR "Reading configuration file $config_file file failed.\n"

    # $OS = "WIN32"; # WIN32 or UNIX
    # $version = "Website jMap Builder 4.1.1";
    while (<CONF>) {
        chomp();
        if ( /POP3 Server =\s+\$1\$/) { $ServerName = $1;}
        elsif ( /POP3 User\s+=\s+\$1\$/) { $UserName = $1;}
        elsif ( /POP3 Password\s+=\s+\$1\$/) { $Password = $1;}
        elsif ( /Request Subject\s+=\s+\$1\$/) { $RequestKey = $1;}
    }
    close(CONF);
    print STDERR "$ServerName:\n";
    print STDERR "User: $UserName\n";
    print STDERR "Password: $Password\n";
    # Email configuration to automate sending jMap to user
}

sub write_log {
    # write log file: timestamp: $msg
    my $msg = shift();
    open (POPLOG, ">>$pop3srv_logfile") || return;
    print POPLOG get_timestamp(), " : $msg\n";
    close (POPLOG);
}

sub get_timestamp{
    # return a time stamp with format: 2002-2-28 15:22:23
    localtime(time);
    my ($sec,$min,$hour,$day,$mon,$year,$wday,$yday,$isdst) =
    localtime(time);
    return "$year-$mon-$day $hour:$min:$sec";
    my $ts = localtime();
    # Sat Feb  9 21:06:56 2002
    if ($ts =~ /\(\w+\)\s+\(\w+\)\s+\(\d+\)\s+\(\d+\)\s+\(\d+\)\s+\(\d+\)/) {
        $ts = "$7-$2-$3 $4:$5:$6";
    }
    return $ts;
}
A-4 PERL/Tk GUI

A-4.1 gui.pl

# Revision History
# Version Date Changes
# 3.0 21-08-00 Add stop, wait, style function to allow use stop
# at any time
# 3.1 5-09-00 Add token number in the processing param line
# 4.0 5 03 01 allow multiple mapping
use Tk;

my $root = "";
my $level=5;
my $max_url =210;
my $join_chk=0;
my $email_chk =0;  # email notification of the jMap is turned off in GUI
my $num_root =0;
my $idchk =0;
my $version = "Website jMap Builder 4.2.0";

# messages
my $readme = "Please fill in the entry fields and click start\nIn $version, you do not need to prefix URL with "http:///\n";
my $msg_warning = "Microsoft Excel cannot handle more than 234
columns\nPlease choose a maximum number of url not more than its limit.";
my $msg_waiting = "Starting retrieving information from site, please
click view log and wait.\n"
my $msg = $readme;
my $command = "";

# file names
my $jmap_path = &getJmapHome();
my $jmap_tmp_file = "$jmap_path/jmaps/"."jmap.txt";
my $report = "$jmap_path/jmaps/"."jmapReport.txt";
my $log = "$jmap_path/tmp/"."jmap.log";
my $stop_file = "$jmap_path/conf/"."Stop.txt";
my $command_file = "$jmap_path/conf/"."commands.txt";
my $session_file = "$jmap_path/conf/"."session.txt";
my $jmap_exe = "$jmap_path/bin/"."webjmapg.pl";
my $jmap_style =1;
my $receiver = "Phillip.Peng@tic.siemens.ca";

if (!-e "$jmap_tmp_file") { unlink "$jmap_tmp_file");
  if (!(-e "$log"))
    { open(LOG, ">$log");
      print LOG "$readme\n";
      close(LOG);
    } }

$smwin = MainWindow->new();
$mwin->title($version);
$menu = $mwin->Menu();
$menu->add('command', -label=>'File');
$lbl_text1 = $mwin->Label(-text => "$version\nCopyright\textcopyright\ 1999-2000 by
Gen-strategies",
-background => 'orange',
-foreground => 'blue',
-anchor => 'n',
-relief => 'groove',
-width => 60,
-height => 2
);

$lbl_text1->pack();

#divide GUI to four frames
$fm_top = $mwin->Frame()->pack(-side => 'top', -fill=>'both');
$fm_left = $mwin->Frame()->pack(-side => 'left', -fill=>'both');
$fm_right = $mwin->Frame()->pack(-side => 'right', -fill=>'both');
$fm_btm = $mwin->Frame()->pack(-side => 'bottom', -fill=>'both');

#two frames in top frame
$fm_top_left = $fm_top->Frame()->pack(-side => 'left', -fill=>'both');
$fm_top_right = $fm_top->Frame()->pack(-side => 'right', -fill=>'both');

#TOP_LEFT TOP & BTM FRAMES
$fm_top_left_top = $fm_top_left->Frame()->
  pack(-side => 'top', -fill=>'both');
$fm_top_left_btm = $fm_top_left->Frame()->
  pack(-side => 'bottom', -fill=>'both');

#URL
$lbl_root = $fm_top_left_top->Label(text => 'Site URL: ',
  relief => 'groove',
  width => 10,
  height => 2)
  ->pack(-side => 'left');

$etr_root = $fm_top_left_top->Entry(-textvariable => \$root,
  -width=>50,
  -background => 'white')
  ->pack(-side => 'left');

#Session ID
$lbl_session = $fm_top_left_btm->Label(text => 'Session ID: ',
  relief => 'groove',
  width => 11,
  height => 2)
->pack(-side => 'left');

$ssidx = &get_sid();
$etr_session = $fm_top_left_btm->Entry(-textvariable => $ssidx,
  -width=>5,
  -state=>'disabled',
  -background => 'white')
->pack(-side => 'left');
$lbl_level = $fm_top_left_btm->Label(-text => 'Max Level: ',
  -relief => 'groove',
  -foreground => 'blue',
  -background => 'white')
-width => 12,
-height => 2
)
->pack(-side => 'left');

$setr_level = $fm_top_left_btm -> Entry( -textvariable => \$level,
 -width=>2,
 -background => 'white')
->pack(-side => 'left');

$lbl_mx_url = $fm_top_left_btm ->Label(text => 'Max URLs: ',
 relief => 'groove',
 width => 10,
 height => 2
 )->pack(-side => 'left');

$etr_mx_url = $fm_top_left_btm -> Entry( -textvariable => \$max_url,
 -width=>3,
 -background => 'white')
->pack(-side => 'left');

$lbl_isJoin = $fm_top_left_btm ->Label(text => 'In Group: ',
 relief => 'groove',
 width => 10,
 height => 2
 )->pack(-side => 'left');

$chk_isJoin = $fm_top_left_btm -> Checkbutton(-variable => \$jcin_chk)
 ->pack(-side => 'left');

#TOP FRAME RIGHT BUTTONS
$addButton = $fm_top_right ->Button(text => 'Add',
 command => \&add_root)
 -> pack(-side=>'left');

$delButton = $fm_top_right ->Button(text => 'Delete',
 command => \&del_root)
 -> pack(-side=>'left');

$button = $fm_top_right ->Button(text => 'Start',
 -state => 'disabled',
 command => \&change_label)
 -> pack(-side=>'left');

# $fm1 = $mwin ->Frame()-pack(-side => 'right');
$lstBox = $fm_left ->Listbox( -height=>20,
 -selectmode => 'multiple',
 -width =>40)
 ->pack(-side=>'left');

$chk_t = $fm_right ->Scrolled("Text", -scrollbars => 'se');

$chk_t ->pack(-expand=>1);
$chk_t -> insert('end', \$text);
$rpt_id = $chk_t -> repeat(500, \&refresh_txt);
MainLoop();

sub refresh_txt {
    $txt_t -> delete('1.0', 'end');
    if (!($button->cget('text') eq "Stop" || $button->cget('text') eq "View Report" || $button->cget('text') eq "Wait")
    }

    $msg = "";
    open(LOG, "$log") || return;
    while (<LOG>)
    {
        $msg .= $_;
        if (($(msg =~ /click View report/) &
            ($button->cget('text') eq "Stop"
            || $button->cget('text') eq "View Report"
            || $button->cget('text') eq "Wait")
        }

        $button -> configure( -text => 'View Report');
    }
    elsif ($(msg =~ /Please start again/) &
            ($button->cget('text') eq "View Report"
            || $button->cget('text') eq "Stop")
    }
    $button -> configure( -text => 'Start');
    }
    close(LOG);
}

$txt_t -> insert('end', $msg);
# $txt_t -> yviewScroll(2, "units");
$txt_t -> Subwidget("yscrollbar")
    -> configure(-background => 'blue');
# $txt_t -> yviewMoveto(1);
}

sub change_label {
    my $stextin = "Hello, this is jmap";
    my $cmds = "";

    if ($button->cget('text') eq "Start")
    {
        $addbutton->configure(-state=>'disabled');
        $delbutton->configure(-state=>'disabled');
        $save_sid($sidx);
        if ($max_url > 250)
        {
            $txt_t -> delete('1.0', 'end');
            $msg = $msg_warning;
            $txt_t -> insert('end', $msg);
            $button -> bell();
        }
        else


{ my $allCmds = $lstBox->get(0, enc);
    open(CMD, ">$command_file");
    for ($j=0; $j<scalar(@allCmds); $j++)
    {
        print CMD "mailto: $receiver
";
        print CMD "$allCmds[$j]
";
    }
    close(CMD);
    $txt_t -> delete('1.0', 'end');
    $msg = $msg_waiting;
    $txt_t -> insert('end', $msg);
    $button->configure(text => 'Stop');
    open(LOG, ">$log");
    print LOG "Start processing urls.
";
    print LOG "The following url will be explored:
";
    print LOG $cmds;
    close(LOG);
    open(FH, "perl $map_exe ");

    }  

    #=head1
    elsif ($button->cget('text') eq "Stop")
    {
        $txt_t -> delete('1.0', 'end');
        my $msg = "STOP=1"
        open(STOP, ">$stop_file") || return;
        print STOP $msg;
        close(STOP);
        $txt_t -> insert('end', "\n\n===Now stopping....\n"");
        open(LOG, ">$log") || return;
        print LOG "\n\n===Now $version is trying to stop the engine,
please wait for....\n\ntone more page being processed.\n\n";
        close(LOG);
        $button->configure(text => 'Wait');
    }

    #=cut

    elsif ($button->cget('text') eq "View Report")
    {
        $txt_t -> delete('1.0', 'end');
        $msg = "";
        open(RP, "$report");
        foreach (<RP>) {
            $msg .= $;
        }
        close(RP);
        $txt_t -> insert('end', $msg);
        $rpt_id -> cancel();
        $button->configure(text => 'Exit');
    }
    elsif ($button->cget('text') eq "Exit")
    {
exit;
}

sub get_sid {
    #$sessionfile = "session.txt";
    open(SESS, "<$session_file") || die("Cannot open session file for validation.
"n");
    my $sess_id=0;
    while (<SESS>){
        chomp();
        if ($_ =~ /^\d+/) {  # $mysid is a local variable
            $sess_id = $1;
            break;
        } # end else
    } # end while
    close(SESS);
    # print join("\n", $sess_id);
    return $sess_id;
}

sub save_sid {
    my $sid3 = shift(@_);
    $sid3 += 1;
    open(SESSL, ">$session_file") || die("unable to open session.txt to overwrite
"n");
    print "\nThis is the new sid: ", $sid3;
    print SESSL "$sid3\n";
    close(SESSL);
}

sub add_root {
    if ($root eq "") { return; }
    $button->configure(-state=>'active');
    $lstBox->activate($num_root++);
    $lstBox->insert($num_root,
        "$sidx\thttp://$root/t$level/t$max_url/t$map_style/t$join_chk/t$email_c
\tik"");
    $lstBox->selectionAnchor($num_root);
    $lstBox->see ($num_root);
    $sidx++;
    $root ="";
    $level=5;
    $max_url =210;
    $join_chk=0;
}

sub del_root {
    my @selectedItems = $lstBox->curselection();
    for ($i=0; $i<scalar(@selectedItems); $i++)
    {
        $lstBox->delete(active); #$selectedItems[$i]
    }
    $lstBox->selectionAnchor(0);
    $lstBox->see(active);
my @allCmds = $lstBox->get(0, end);
if (scalar(@allCmds) == 0)
{
    $button->configure(-state=>'disabled');
}

sub getJmapHome{
    use Cwd;
    chdir("..");
    my $home = getcwd();
    chdir("bin");
    return $home;
}

A-5 Shell Batch Files

A-5.1 DOS gui.bat

perl gui.pl

A-5.2 DOS jmap.bat

perl gui.pl

A-5.3 UNIX gui

#!/usr/bin/sh
perl gui.pl

A-5.4 UNIX jmap

#!/usr/bin/sh
perl gui.pl
Appendix B: An Example of Web Site jMap

Model

The following section shows a hard copy of the recovered jMap web site model by this program from www.cs.concordia.ca. The total 45 URLs have been retrieved and analysed to build this web site jMap model as an example.