Digital library logging and analysis using XML

CS 5974 Independent Study Report
Spring 2003

Ganesh K. Panchanathan
226-91-3494
gpanchan@vt.edu
Instructor: Dr. Edward Fox

Table of contents

1. Introduction

Log analysis is the primary source of information about how digital library (DL) patrons interact with the system and the way the DL responds to patrons' requests for information [1]. Hence, log recording plays an important role in evaluating the existing services of a digital library, enabling improvements and enhancements for the same. Although current digital libraries based on web servers perform logging at the HTTP layer, there is scope for having a better logging scheme built into the digital library. The new logging scheme will differ from the existing logging method in both the information that is being logged and the format in which it is logged [1]. The new logging scheme will promote interoperability, better analysis, and reuse of log analysis tools.

This report lists and documents the work done in the field of digital library logging using XML over the last year at Digital Library Research Laboratory, Virginia Tech (DLRL). The starting point for the project was the logging standard proposed by Dr. Edward Fox and Marcos Goncalves to use XML for logging in digital libraries [1]. This log format captures a rich, detailed set of system and user behaviors supported by the current digital libraries. The DL log file proposed by this standard contains log entries with each entry representing a type of user action.

1.1 Motivation for digital library logging in XML

The MARIAN digital library was taken as a case study to analyze the way logging was done in an existing DL system. After code analysis, the following reasons were identified as to the motivation for redesigning the logging system of the digital library.

1. To enable loose coupling of the logging module in the destination DL system such as MARIAN or CITIDEL.
2. If the logging module has a single point of interaction with the DL system, it will be easier to upgrade the logging system.
3. To avoid recompilation of the whole DL system whenever the logging schema or the logging module needs to be changed.
4. In the old implementation, the logging instructions were spread out within the DL source code. The logging module will integrate the functionality into a single cohesive component.

1.2 Goals of the independent study

1. Integrate the XML logging module with CITIDEL and perform testing.
2. Design tools for performing logging analysis on the collected logs.
3. Tools to generate statistics and visualizations from the above analysis methods.
4. Maintain DL Log format and make necessary modifications for any new efforts.
5. Document the log format and the logging module for future work.

2. XML Logging Standard and API

2. 1 XML Logging Standard

The XML logging proposed standard describes a set of user and system behavior supported by current digital libraries. The standard defines an extensive set of attributes and arranges them in a logical and hierarchical fashion. The standard is defined using XML Schema. XML Schema provides a means for defining the structure, content, and semantics of XML documents [4]. The Schema can be revised, so this is an extensible standard; the description in this report reflects its current status.

The logging standard supports a session based digital library system. A session includes all the events taking place after the user has logged in and before the user logs out or times out. Transactions are categorized as related to user and system events associated with the use of the DL services, session creation, user registration, administration activities, errors, and user responses [1]. Each transaction is associated with a statement.

The statements could be any of the following:

a. Session Info
b. Registration info
c. Admin info
d. Error info
e. Help info
f. Event

Each event is associated with an action. Currently, four actions are defined. They are:

a. Search
b. Browse
c. Update
d. Store system information.

Thus the proposed XML log standard defines the schema for an interoperable, reusable schema for digital library logging. The schema for the logging standard is listed in Appendix 6.1.

2.2 API for XML Logging in digital libraries

The logging API was developed using the JAVA language since JAVA is platform independent. It enabled the logging API to be compatible and deployable in digital libraries implemented in any platform for which the Java runtime exists.

2.2.1 Features of the logging API

1. Hierarchical – The hierarchy of the classes in the API resembles the hierarchy of elements in the log schema.
2. Bottom-up Approach – The log record is built in a bottom-up fashion, i.e., the lower level objects are built first, which in turn are used to build upper level objects.
3. XML formatting at each level – The information gathered within the objects at each level is formatted into XML.

4. Comprehensive – The logging schema attempts to encompass the broad range of information that can be collected about user behavior and system actions. If the DL system does not furnish a part of the required information, then the particular fields will not be output to the XML log.

2.2.2 Testing of the API

Testing of the API was done using test drivers. The test drivers aim to simulate the calls from the destination digital library system.

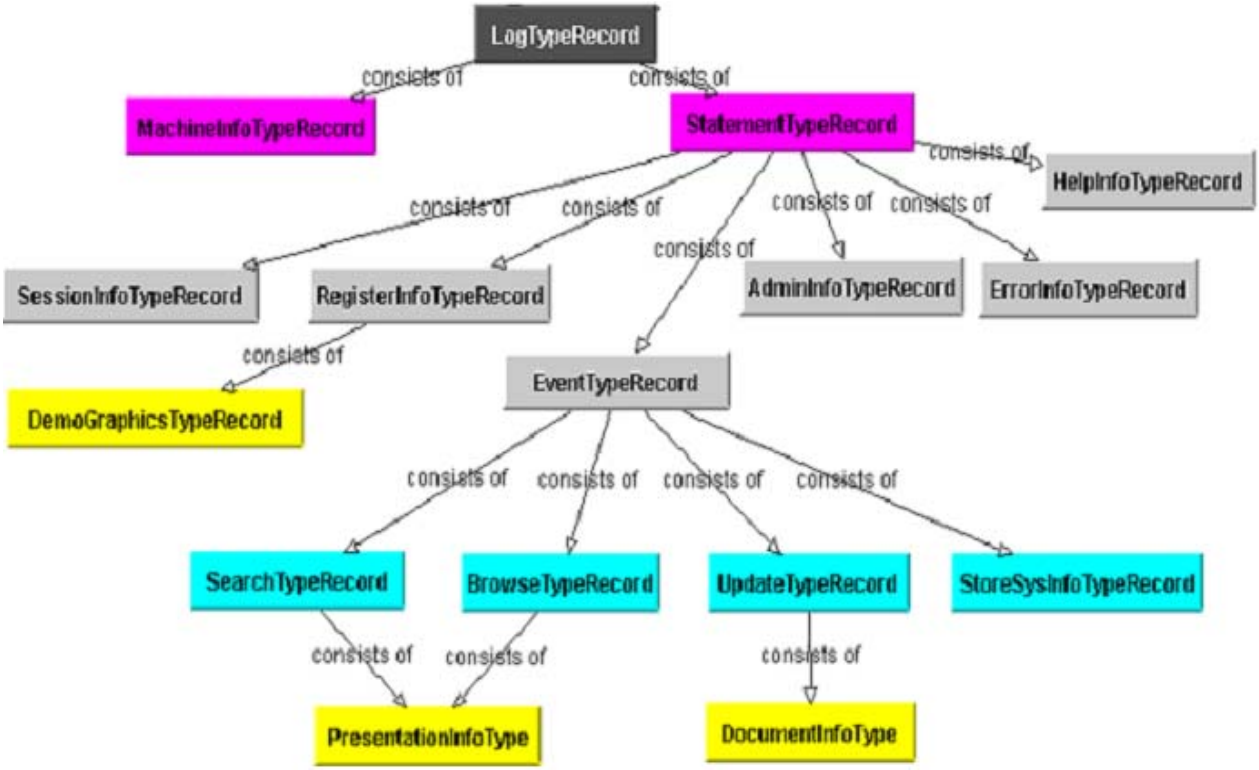
A set of scenarios was devised. The scenarios deal with:

1. User logging in
2. User searching for a document
3. User browsing a document
4. User logging off

Two kinds of testing were done on the API:

1. Unit testing was done for the individual classes.
2. Integrated testing was done to verify that the different classes worked together without any problems

2.2.3 Concept map for the logging API:



3. Integration of XML Logging in CITIDEL

3.1 Introduction

CITIDEL (Computing and Information technology interactive digital educational library) aims to establish, operate, and maintain a part of the NSDL that will serve the computing education community in all its diversity and at all levels [5]. This will include computer science, information systems, information science, software engineering, and computer engineering.

3.1.1 Goals

The goals for integrating the logging module into CITIDEL are listed below:

1. Enable logging of user behavior and system actions in the CITIDEL system.
2. Test the deployment of the logging API in yet another DL system to identify and understand the problems if any.
3. To be the first attempt at achieving distributed logging where the logging client and the logging module are on different machines or different virtual machines.

3.1.2 Challenges

1. CITIDEL was a PERL based system whereas the logging API was implemented in Java.
2. One way to establish communication between a PERL and Java module was to use JNI (Java Native Interface). But this method had several restrictions.
3. The current implementation uses sockets to communicate between the CITIDEL system and the logging module. This implementation is efficient due to the fact that sockets are language and platform independent.

3.2 Implementation

3.2.1 Approach

1. The first step was to identify the attributes that can be logged for the CITIDEL system. Although the Schema encapsulates many attributes, CITIDEL will require and log only a subset of the attributes.
2. A simple protocol was developed for communication between CITIDEL and the logging module. The protocol is explained in section 3.2.3.
3. A logging interface was developed that would receive logging requests from CITIDEL running on a remote machine and invoke appropriate calls on the logging API so that the logs are collected on the local machine.
4. After the interface was developed, it was bundled with the API to be sent to the partners who expressed interest in the logging system.

3.2.2 Communication

For communication between the CITIDEL client and logging module, connectionless sockets were used. Logging information was sent in UDP packets since the transfer was one way from client to the logging module. The logging module listens at port 8888.

3.2.3 Protocol

Each datagram packet should follow the following pattern:

#IDENTIFIER \n
INFO1 \n
INFO2 \n
INFO3 \n
…. \n
…. \n
\n

where:

#IDENTIFIER can be any of:
#LOGIN : when a user logs in
#LOGOUT : when an user logs out
#SEARCH : when a user searches for documents in DL
#BROWSE: when a user browses through documents in DL
#EXIT: when the DL system exits
\n : new line character

The packet formats for the different commands are listed below:

#LOGIN	#SEARCH	#BROWSE	#LOGOUT
SessionID	Session ID	Session ID	SessionID
LoginID	LoginID	LoginID	LoginID
Timestamp	Timestamp	Timestamp	Timestamp
	QueryString	DocID	
	SearchBy (Author, Keywork, All, etc.)	DocumentName	
		CollectionName	
	CollectionName	BytesTransferred	
	NoOfDocumentsFound		
	HighestMatchDocName		
	LowestMatchDocName		

A sample set of protocol commands are listed in Appendix 6.2.

3.2.4 Configuration file

The logging interface requires a configuration file, which contains the various parameters required for setting up the logging interface for proper functioning. The configuration parameters and their values are listed below:

KEYWORD	VALUE
Schema_Loc	Location (Path and filename) of the schema for the DL Log format.
Log_Loc	Location (Path) of the destination log file.
Log_Fname_Format	FIlename format of the destination log file. Log Filename format includes: "yr" = year "mh" = month "dy" = day "hr" = hour "mn" = minute "sc" = second
Port_Num	Port Number at which the LoggerModule will receive the commands.
Log_Rotation_Mode	Criteria for log rotation. This might be one of the following: "FILE_SIZE" Log is rotated when the size of the log file exceeds Log_Rotation_Limit KBs. "EVENT_COUNT" Log is rotated when Log_Rotation_Limit count of events are logged.
Log_Rotation_Limit	Limit for log rotation. Can be KiloBytes of logfile size or count of events logged.

An example configuration file is listed in Appendix 6.4.

4. Log Analysis

4.1 Introduction

To utilize the full advantage of XML Logging, the collected log files must be analyzed and the results of the analysis must be tabulated or presented in an intuitive form. This would help the administrators of the digital library to discover patterns in user behavior in order to improve the user experience while using the digital library. This also would help them to identify potential problems in resource availability and rectify them before it becomes a major bottleneck for users accessing information from the digital library.

4.1.1 User activity files

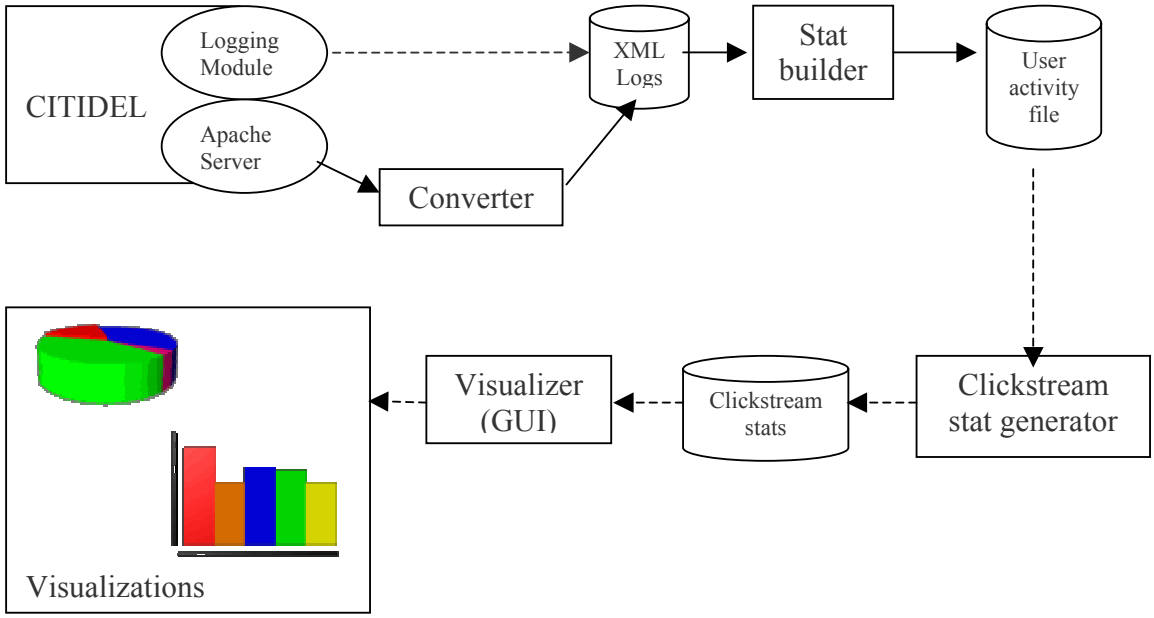
The starting point for the visualization efforts is the XML Log Analyzer [2] built by Filip Jagodzinski of Villanova University. The XML Log Analyzer produces viable user statistics that will aid the evaluation process and help the digital library development team to produce the best-possible digital library interface [2]. The user activity files contain a set of intermediate statistics generated by analyzing the XML logs. The following are the different types of user activity files:

1. UserID – information about users and number of times they logged in.
2. DocID – information about the accesses made for each document.
3. IPID – information about IP addresses from which accesses take place.

4.1.2 Steps to be taken

1. Brainstorming and coming up with a set of statistics that are of utmost importance for analyzing user behavior in digital libraries.
2. Explaining the statistics and providing valid justifications for collecting them.
3. Generation of statistics:
   Identify the relevant statistics from the user activity files and process them in order to convert them to visualizations. Processing normally would involve aggregating the available data into concise distributions across domains, time, etc.
4. Generation of Visualization code:
   After the statistics are generated, they are converted into PHP files containing instructions to produce visualizations. In order to accomplish this, we used the JpGraph library which is an OO Graph drawing library for PHP. The PHP file can be hosted on a web server and it will dynamically produce the visualization when a user requests it from a browser.
5. Documenting all the work done including the statistics and the implementation code.

4.1.3 Overall architecture



This figure shows the overall architecture of the logging process in CITIDEL and where this project fits in. Right now, XML Logs are collected from CITIDEL using a converter that converts Apache logs to XML. In the future, a logging module will be added to CITIDEL that would output XML Logs. Filip's work involved parsing the XML Logs and converting them into User activity files which are explained in 1.1.1. Then the User activity files are converted into Clickstream stats. These statistics are then used to generate visualizations such as bar graphs and pie charts.

4.2 Potential Statistics

A set of potential statistics that can be collected are listed below. Out of these, three statistics were selected for implementation. Visualization methods were also designed and implemented for the statistics. The prototypes for statistics extraction and visualizations are explained in section 4.3.

4.2.1 Session Based:

a. Number of searches per download

For each download that is logged, we will search backwards and compute the number of searches that occurred between downloads. Upon reaching the next download logged, the search would stop. This information will be valuable in analyzing the efficiency and performance of the search engine in returning useful results to the user.

b. Number of browses per download

A user browses when he/she goes through the different categories and sub-headings in which the documents are arranged. This statistic measures how frequently the user downloads documents while browsing. This is measured by counting the number of browses between downloads.

c. Number of downloads per search

Between successive searches, we count the number of downloads performed by the user. This information is useful in that it shows the number of results deemed pertinent by the user during a single search.

d. Total time per session

The time between login and logout is measured to account for user's total session time. This statistic is valuable because it shows the amount of time that the user utilizes the digital library. In cases when the user fails to logout, an inactivity timer is used to identify user time-outs.

e. Number of searches per session

The number of searches that the user performs each session is recorded in order to show the general efficiency of the user's utilization of the digital library.

f. Number of downloads per session

The number of downloads that the user performs each session is recorded in order to measure how much information the user obtains from the session.

4.2.2 Time Based:

a. Time between search and first download

The difference between the time of a search and the time of the first subsequent download is recorded in this field. Said information will determine the amount of time the user allocates to browsing search results and revising the search.

b. Time spent for each browse

This is measured by subtracting the time of the browse from the time of the next action performed by the user, whether it is a download, a search, or a logout. Such information is beneficial in gauging the general effectiveness of abstracts and category indices in describing a library entry to the user.

c. Time between searches

Computed by subtracting the timestamp on a search from that of the immediately previous search, this statistic is valuable because it accurately describes the amount of time that the user takes in viewing search results, browsing results, or downloading.

4.2.3 Distribution of Time Based

Statistics about users and their actions can be distributed across the time of the day. For this purpose, the time of the day can be divided into four different zones. They are 12:01–6:00 AM, 6:01–12:00 PM, 12:01–6:00 PM and 6:01–12:00 AM.

a. Number of users

The total number of users for a given period is tracked for bandwidth, geographic-use, and server-maintenance information.

b. Number of downloads

The number of downloads for a given period is recorded to show the time period in which most users are actively retrieving information from the digital library.

c. Number of searches

The number of searches for a given period is also recorded to show the time period in which most users are actively seeking information on the digital library.

4.2.4 Distribution of Domain Based (.edu / .com / .org / .gov / etc.):

a. Number of users

The number of users from a specific domain field is computed during a time period to ascertain the amount of traffic provided by a given group of users.

b. Number of downloads

The number of downloads from a specific domain shows the amount of information retrieved by users from a given background.

c. Number of searches

The number of searches from a specific domain shows the amount of information sought by users from a given background.

4.2.5 Word Based:

a. Percentage of words repeated across search refinements

The percentage of words repeated across search refinements is calculated by dividing the number of similar words between successive searches by the total number of words in the refined search (e.g., Search 1 “A B C”, Search 2 “A B D E”, Percentage = 2 / 4 = 50%). This statistic gives us an idea of the type of information for which the user is searching as well as demonstrating the ability of the user to find valued information.

b. Top three words used across searches in a session

This statistic involves the three highest number of words repeatedly used within a session. In the event of a tie between words, the word that is alphabetically lower is taken. This information is useful, because it contains what users are looking for when they visit the digital library.

4.3 Sample statistics and visualizations

4.3.1 Domain distribution

The goal is to identify the top 5 domains from which most accesses take place and generate a table containing the domains and the total hits. This statistic can be best visualized using a pie chart showing the % of accesses from these domains. The required statistic is generated from the "IPID" user activity file. The file contains data in the following format:

IP_ID <hostname> Hits <# of hits>

Example:

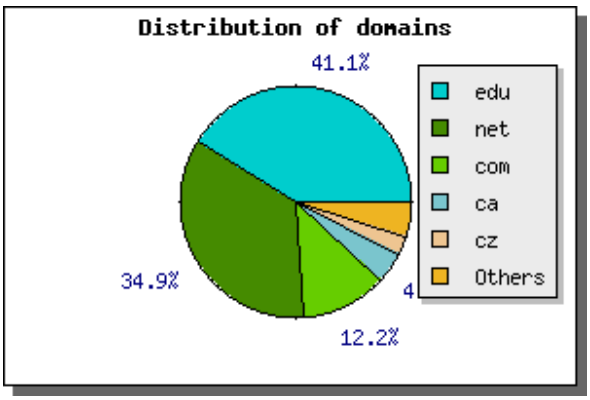
IP_ID 216-12-11-140.dmt.ntelos.net Hits 000461
IP_ID ip68-12-76-50.ok.ok.cox.net Hits 000003
IP_ID 214.143.226.200.in-addr.arpa.ig.com.br Hits 000002
IP_ID crawler11.googlebot.com Hits 000038

The file is read and converted to the following aggregate statistic containing the distribution of the accesses across the domains:

Login count interval	**# of logins**
Edu	1278
Net	1087
Com	378
Ca	131
Cz	83
others	154

This contains the top 5 domain names and the number of accesses from each of the domains. All the accesses from other domains are added into the "Others" category.

The next step is the generation of PHP code that contains JpGraph objects. The above values viz. the domain names and the number of accesses must be included in the PHP template. For this example, a pie chart will be produced that will show the distribution of accesses across domains. The following pie chart is generated for the above example:



4.3.2 Hit distribution

The goal of this visualization is to display how many times documents are accessed, highlighting the most. The frequency of hits for each document is obtained from the "DocID" user activity file. It contains information about accesses in the following format:

DocID <ID> Hits <total hits> EduHits <#> GovHits <#> ComHits <#> NetHits <#> OrgHits <#> OtherHits <#>

Example:

DocID Browse_Scheme_CCS1998 Hits 000159 EduHits 000041 GovHits 000000 ComHits 000018 NetHits 000040 OrgHits 000000 OtherHits 000059

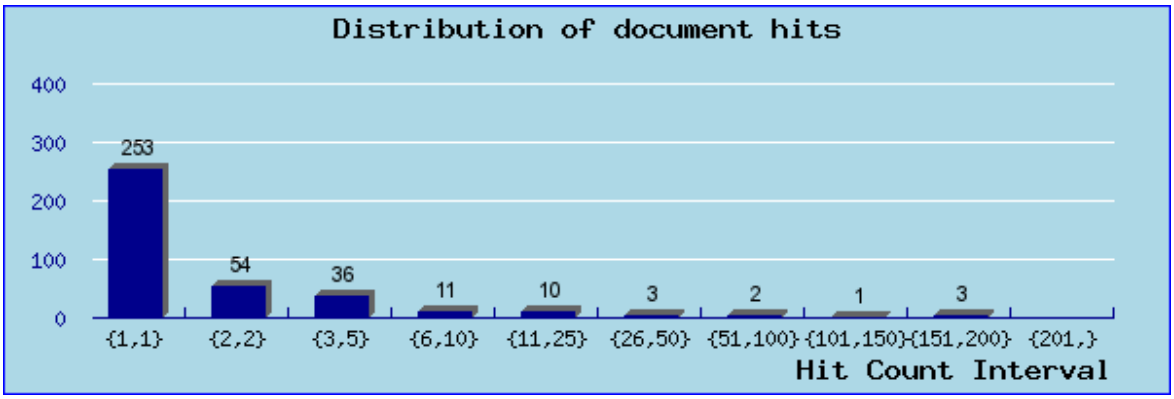
DocID Browse_Scheme_CCS1998&node=198 Hits 000034 EduHits 000012 GovHits 000000 ComHits 000004 NetHits 000013 OrgHits 000000 OtherHits 000004

DocID Browse_Scheme_CCS1998&token=35262& Hits 000003 EduHits 000000 GovHits 000000 ComHits 000000 NetHits 000002 OrgHits 000000 OtherHits 000000

Data from this file is analyzed and the number of documents falling between a set of hit count intervals are calculated. Example of the generated statistic is as follows:

Hit count interval	**# of hits**
{1,1}	253
{2,2}	54
{3,5}	36
{6,10}	11
{11,25}	10
{26,50}	3
{51,100}	2
{101,150}	1
{151,200}	3
{201,-1}	0

This data is visualized as a bar chart with the hit count intervals on the X axis and the # of hits on the Y axis. An example bar chart for the above data is shown below:



4.4.3 Login distribution

The goal of this visualization is to display how many times users login to the digital library system. The frequency of logins for each user is obtained from the "UserID" user activity file. It contains information about accesses in the following format:

UserID <USER> Logons <#>

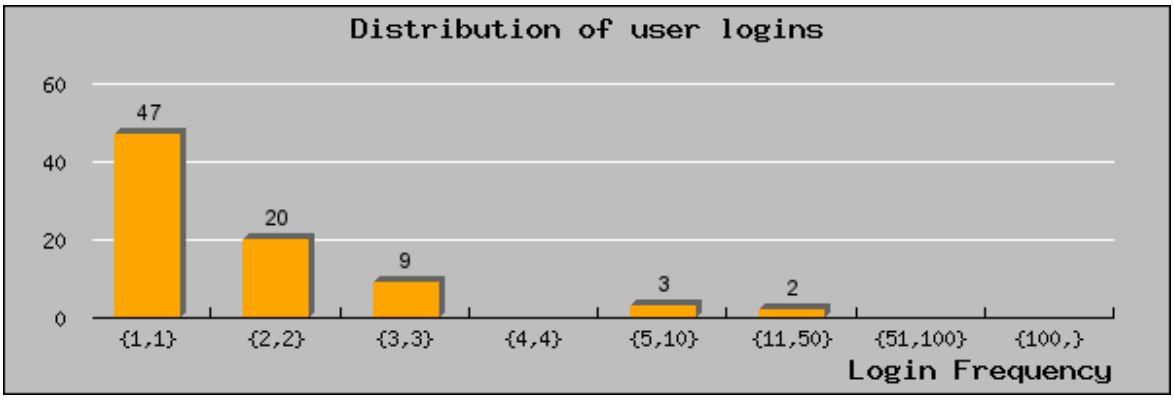
Example:

UserID Converted_Log_64.68.82.70 Logons 000003
UserID Converted_Log_64.68.82.67 Logons 000002
UserID Converted_Log_216.12.11.140 Logons 000025
UserID Converted_Log_66.208.16.30 Logons 000001
UserID Converted_Log_198.82.191.240 Logons 000001

Data from this file is analyzed and the number of users falling into a set of login count intervals is calculated. An example of this statistic is as follows:

Login count interval	**# of logins**
{1,1}	47
{2,2}	20
{3,3}	9
{4,4}	0
{5,10}	3
{11,50}	2
{51,100}	0
{101,}	0

This data is visualized as a bar chart with the login frequency on the X axis and the # of logins on the Y axis. An example bar chart for the above data is shown below:



5. References

[1] *An XML Log Standard and Tool for Digital Library Logging Analysis*, Marcos André Gonçalves, Ming Luo, Rao Shen, Mir Farooq, and Edward. A. Fox. Proceedings of the Sixth European Conference on Research and Advanced Technology for Digital Libraries, Rome, Italy, September 16-18, 2002

[2] *XML Log Analyzer, CITIDEL Digital Library,* Filip Jagodzinski, Villanova University, http://www.csc.vill.edu/~fjagodzi/CITIDEL_work/LogAnalyzer/design.html, April 05, 2003

[3] *The XML Log Standard for Digital Libraries: Analysis, Evolution, and Deployment*, Marcos André Gonçalves, Ganesh Panchanathan, Unnikrishnan Ravindranathan, Aaron Krowne, Edward. A. Fox, Filip Jagodzinski, and Lillian Cassel, Third Joint Conference on Digital Libraries, Houston, Texas, May, 2003

[4] *XML Schema,* http://www.w3.org/XML/schema, March 20, 2003

[5] *CITIDEL,* http://www.citidel.org/about.html, February 10, 2003

6. Appendices

6.1 XML Log schema

The following schema defines the syntax and semantics for the XML logs that are collected from CITIDEL. The log schema is written using XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:element name="Log" type="LOGType"/>
   <xsd:complexType name="LOGType">
      <xsd:sequence>
         <xsd:element name="LogEntry" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="TimeStamp" type="xsd:string"/>
                  <xsd:element name="Transaction" type="xsd:string"/>
                  <xsd:element name="SessionInfo" type="SessionInfoType" minOccurs="0"/>
                  <xsd:element name="MachineInfo" type="MachineInfoType" minOccurs="0"/>
                  <xsd:element name="Statement" type="StatementType" minOccurs="0"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="MachineInfoType">
      <xsd:sequence>
         <xsd:element name="IPAddress" type="xsd:string"/>
         <xsd:element name="Port" type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="StatementType">
      <xsd:sequence>
         <xsd:element name="RegisterInfo" type="RegisterInfoType" minOccurs="0"/>
         <xsd:element name="Event" type="EventType" minOccurs="0"/>
         <xsd:element name="AdminInfo" type="AdminInfoType" minOccurs="0"/>
         <xsd:element name="ErrorInfo" type="ErrorInfoType" minOccurs="0"/>
         <xsd:element name="HelpInfo" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="SessionInfoType">
      <xsd:sequence>
         <xsd:element name="SessionID" type="xsd:string"/>
         <xsd:element name="SessionStartTime" type="xsd:string"/>
```

```
      <xsd:element name="LoginID" type="xsd:string"/>
      <xsd:element name="UserInfo" type="xsd:string" minOccurs="0"/>
      <xsd:element name="SessionEndTime" type="xsd:string" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="RegisterInfoType">
   <xsd:sequence>
      <xsd:element name="LoginID" type="xsd:string"/>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="PrivilegesInfo" minOccurs="0">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="Domain" type="xsd:string" minOccurs="0"/>
               <xsd:element name="BibList" type="xsd:string" minOccurs="0"/>
               <xsd:element name="Group" type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="Address" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Phone" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Email" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Fax" type="xsd:string" minOccurs="0"/>
      <xsd:element name="DemographicsInfo" type="DemographicsType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="EventType">
   <xsd:sequence>
      <xsd:element name="EventString" type="xsd:string"/>
      <xsd:element name="Action">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="Search" type="SearchType" minOccurs="0"/>
               <xsd:element name="Browse" type="BrowseType" minOccurs="0"/>
               <xsd:element name="Update" type="UpdateType" minOccurs="0"/>
               <xsd:element name="Other" type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="StatusCode" type="xsd:int"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AdminInfoType">
   <xsd:sequence>
      <xsd:element name="BackupFile" type="xsd:string" minOccurs="0"/>
      <xsd:element name="BackupStart" type="xsd:string" minOccurs="0"/>
      <xsd:element name="BackupEnd" type="xsd:string" minOccurs="0"/>
```

```
      <xsd:element name="SystemStart" type="xsd:string" minOccurs="0"/>
      <xsd:element name="SystemEnd" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="ErrorInfoType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="InvalidQuery"/>
      <xsd:enumeration value="ObjectNotFound"/>
      <xsd:enumeration value="NotAcessible"/>
      <xsd:enumeration value="ConnectionTimedOut"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="DemographicsType">
    <xsd:sequence>
      <xsd:element name="Gender" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Discipline" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Age" type="xsd:int" minOccurs="0"/>
      <xsd:element name="Ethnicity" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="SearchType">
    <xsd:sequence>
      <xsd:element name="Collection" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ObjectType" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="DigitalObjectType" type="xsd:string" minOccurs="0"/>
            <xsd:element name="HoldingRecords" type="xsd:string" minOccurs="0"/>
            <xsd:element name="CommunityRecords" type="xsd:string" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="SearchBy" type="xsd:string" minOccurs="0"/>
      <xsd:element name="SearchType" type="xsd:string" minOccurs="0"/>
      <xsd:element name="QueryString" type="xsd:string" minOccurs="0"/>
      <xsd:element name="TimeFrame" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="StartDate" type="xsd:string"/>
            <xsd:element name="EndDate" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="SearchSysInfo" type="SearchSysInfoType" minOccurs="0"/>
      <xsd:element name="PresentationInfo" type="PresentationInfoType" minOccurs="0"/>
    </xsd:sequence>
```

```
</xsd:complexType>
<xsd:complexType name="BrowseType">
   <xsd:sequence>
      <xsd:element name="DocName" type="xsd:string"/>
      <xsd:element name="DocumentInfo" type="DocumentInfoType" minOccurs="0"/>
      <xsd:element name="BytesTransferred" type="xsd:int" minOccurs="0"/>
      <xsd:element name="PresentationInfo" type="PresentationInfoType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="UpdateType">
   <xsd:sequence>
      <xsd:element name="EditInfo" type="DocumentInfoType" minOccurs="0"/>
      <xsd:element name="AddInfo" type="DocumentInfoType" minOccurs="0"/>
      <xsd:element name="DeleteInfo" type="DocumentInfoType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DocumentInfoType">
   <xsd:sequence>
      <xsd:element name="PathName" type="xsd:string" minOccurs="0"/>
      <xsd:element name="DocId" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Collection" type="xsd:string" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SearchSysInfoType">
   <xsd:sequence>
      <xsd:element name="BytesTransferred" type="xsd:int" minOccurs="0"/>
      <xsd:element name="NoOfDocFound" type="xsd:int" minOccurs="0"/>
      <xsd:element name="OutOfDoc" type="xsd:string" minOccurs="0"/>
      <xsd:element name="ResponseTime" type="xsd:int" minOccurs="0"/>
      <xsd:element name="HighestMatch" type="xsd:string" minOccurs="0"/>
      <xsd:element name="LowestMatch" type="xsd:string" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="PresentationInfoType">
   <xsd:sequence>
      <xsd:element name="Format">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="List"/>
               <xsd:enumeration value="Threaded"/>
               <xsd:enumeration value="Tabular"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="SortBy">
         <xsd:simpleType>
```

```
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="By Author"/>
              <xsd:enumeration value="By Confidence"/>
              <xsd:enumeration value="By Subject"/>
              <xsd:enumeration value="By Date"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="NumberofResults" type="xsd:int" minOccurs="0"/>
        <xsd:element name="Cutoff" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="Confidence"/>
              <xsd:enumeration value="Number of Results"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

6.2 Sample set of protocol commands

The following section lists a set of commands that are sent from the logging client to the logging interface. This communication takes place through sockets. The logging interface parses the command and logs the information sent through the command.

#LOGIN
2340
john
Mon 11/25/2002 18:43:54.33

#SEARCH
2350
smith
Mon 11/25/2002 18:43:54.33
computer algorithms
keyword
CITIDEL
324
firstdoc
lastdoc

#BROWSE
2350
smith
Mon 11/25/2002 18:43:54.33
865554-test-342
djikstra_algorithm.pdf
CITIDEL
13346

#LOGOUT
2345
smith
Mon 11/25/2002 18:43:54.33

6.3 Sample log file

The following section contains a sample log file that was produced when the protocol commands listed in appendix 6.2 were sent to the logging interface.

```
<?xml version="1.0" ?>
<Log xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="/citidel/www.citidel.org/lib/logging/LOG.xsd">
<LogEntry>
        <TimeStamp>Mon 11/25/2002 18:43:54.33</TimeStamp>
        <Transaction>0</Transaction>
        <SessionInfo>
                <SessionID>2340</SessionID>
                <LoginID>john</LoginID>
        </SessionInfo>
        <MachineInfo>
                <IPAddress>127.0.0.1</IPAddress>
                <Port>7777</Port>
        </MachineInfo>
        <Statement>
                <Event>
                        <Action>
                        <Other>Login</Other>
                        </Action>
                </Event>
        </Statement>
        </LogEntry>
<LogEntry>
        <TimeStamp>Mon 11/25/2002 18:43:54.33 </TimeStamp>
        <Transaction>1</Transaction>
        <SessionInfo>
                <SessionID>2350</SessionID>
                <LoginID>smith</LoginID>
        </SessionInfo>
        <MachineInfo>
                <IPAddress>127.0.0.1</IPAddress>
                <Port>7777</Port>
        </MachineInfo>
        <Statement>
        <Event>
                <Action>
                <Search>
                        <Collection>CITIDEL</Collection>
                        <SearchBy>keyword</SearchBy>
                        <QueryString>computer algorithms</QueryString>
                        <SearchSysInfo>
```

```
                    <NoOfDocFound>324</NoOfDocFound>
                    <HighestMatch>firstdoc</HighestMatch>
                    <LowestMatch>lastdoc</LowestMatch>
                    </SearchSysInfo>
                </Search>
                </Action>
        </Event>
        </Statement>
</LogEntry>
<LogEntry>
        <TimeStamp>Mon 11/25/2002 18</TimeStamp>
        <Transaction>2</Transaction>
        <SessionInfo>
                <SessionID>2350</SessionID>
                <LoginID>smith</LoginID>
        </SessionInfo>
        <MachineInfo>
                <IPAddress>127.0.0.1</IPAddress>
                <Port>7777</Port>
        </MachineInfo>
        <Statement>
        <Event>
                <Action>
                <Browse>
                        <DocName>djikstra_algorithm.pdf</DocName>
                        <DocumentInfo>
                        <DocId>865554-test-342</DocId>
                        <Collection>CITIDEL</Collection>
                        </DocumentInfo>
                        <BytesTransferred>13346</BytesTransferred>
                </Browse>
                </Action>
        </Event>
        </Statement>
</LogEntry>
</Log>
```

6.4 Sample Configuration file

The following section lists an example configuration file that was used to supply configuration parameters to the logging interface. It contains parameters such as the naming format for the log file, log rotation rules, etc.

#Configuration settings for digital library logging using XML

#Location of the DL Log schema
Schema_Loc C:\Citidel_Logging\LOG.xsd

#Location of the log file
Log_Loc C:\Citidel_Logging\

#Port number
Port_Num 8888

#Log file name format
Log_Fname_Format Log_$yr_$mh_$dy_$hr_$mn_$sc.xml

#Log Rotation Option FILE_SIZE/EVENT_COUNT
Log_Rotation_Mode EVENT_COUNT

#Log Rotation limit - Size in KB/Number of events
Log_Rotation_Limit 5

6.5 Generated code for PHP visualization

Visualizations are generated in the browser using PHP code. The following is an example of the PHP code generated to display a pie chart showing the domain distribution.

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_pie.php");

// Some data
$data = array(1278,1087,378,131,83,154);
// Create the Pie Graph.
$graph = new PieGraph(300,200,"auto");
$graph->SetShadow();

// Set a title for the plot
$graph->title->Set("Distribution of domains");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Create
$p1 = new PiePlot($data);
$p1->SetLegends(array("edu","net","com","ca","cz","Others"));
$graph->Add($p1);
$graph->Stroke();
?>
```