An XML Log Standard and Tool for Digital Library Logging Analysis

Marcos André Gonçalves, Ming Luo, Rao Shen, Mir Farooq Ali, and Edward A. Fox

Virginia Tech, Blacksburg VA 24061, USA
{mgoncalv,fox}@vt.edu

Abstract. Log analysis can be a primary source of knowledge about how digital library patrons actually use DL systems and services and how systems behave while trying to support user information seeking activities. Log recording and analysis allow evaluation assessment, and open opportunities to improvements and enhanced new services. In this paper, we propose an XML-based digital library log format standard that captures a rich, detailed set of system and user behaviors supported by current digital library services. The format is implemented in a generic log component tool, which can be plugged into any digital library system. The focus of the work is on interoperability, reusability, and completeness. Specifications, implementation details, and examples of use within the MARIAN digital library system are described.

1 Introduction

Log analysis is a primary source of knowledge about how digital library patrons actually use DL systems and services and how systems behave while trying to support user information seeking activities. Log recording and analysis allows evaluation assessment and opens opportunities to improvements and enhanced new services. Indeed, the benefits of logging are numerous, including improving performance by recording effective evaluation data [13], helping in designing and testing of user interfaces [7], and better allocation of resources [17].

Conventional libraries have a long history of concern for privacy [10]. While circulation statistics are widely available, storage of patron-related information is rare in such libraries. The introduction of On-Line Public Access Catalogs (OPACs) has changed the picture and allowed some degree of log recording and analysis to improve library services [1, 16, 17, 15]. More recently, web servers and proxy caching servers have made web log analysis become common place, recording each and every access to their documents. These, along with the advance of techniques in web log mining, have made possible a number of new and enhanced services such as customization and personalization [14].

Digital libraries differ from the Web in many ways. Firstly, digital library collections are explicitly organized, managed, described, and preserved. Secondly, web sites and web search engines assume very little about the users, tasks, and data they deal with. Digital libraries normally have much more knowledge of their users and tasks since they are built to satisfy specific needs of interested communities. And thirdly, the digital objects in DL collections tend to be much more structured than the information presented in the Web. Therefore, digital library logging should offer much richer information and opportunities. Despite the fact that many current DL systems do some kind of logging, they tremendously differ in the format in which they record the information and even the sort of information that is recorded. Interoperability, reuse of log analysis tools, and comparability of log analysis results are major problems.

In this paper, we propose an XML-based standard digital library log format that captures a rich, detailed set of system and user behaviors supported by current digital library services. The proposed standard is implemented in a generic log component tool, which can be plugged into any digital library system to produce the specified format. The focus of this work is on interoperability, reusability, and completeness. Specifications, implementation details, and examples of use within the MARIAN digital library system are described.

This paper is organized as follows. Section 2 covers related work and analyzes associated problems. Section 3 describes the DL log format and motivation for design. Section 4 presents the log tool, its implementation and some examples. Section 5 outlines future work and concludes the paper.

2 Related Work

Most current Web servers store log files in the Common Log Format (CLF)- a simplistic format which reflects the stateless nature of the HTTP protocol by recording just individual server events. Apache, perhaps the most used web server, uses an extension of CLF called Combined Log Format, which tries to keep some state information by recording the links between resources.

A sample of CLF is given below. The fields are host; rfc931, i.e., information returned regarding identity of the person, otherwise '-'; authuser, if a userid is sent for authentication, otherwise '-'; day; month; year; hour; minutes; seconds; request; the first line of the HTTP request as sent by the client; ddd, status code returned by the server, otherwise '-'; and bbb, the number of bytes sent (not including the HTTP/1.0 header), otherwise '-'.

```
bbn-cache-3.cisco.com - - [22/Oct/1998:00:20:21 -0400] "GET
/~harley/courses.html HTTP/1.0" 200 1734
bbn-cache-3.cisco.com - - [22/Oct/1998:00:20:22 -0400] "GET
/~harley/clip_art/word_icon.gif HTTP/1.0" 200 1050
www4.e-softinc.com - - [22/Oct/1998:00:20:27 -0400] "HEAD
/ HTTP/1.0" 200 0
user-38ldbam.dialup.mindspring.com - - [22/Oct/1998:00:20:48 -0400] "GET
/~lhuang/junior/capehatteras.html HTTP/1.0" 200 328
user-38ldbam.dialup.mindspring.com - - [22/Oct/1998:00:20:48 -0400] "GET
/~lhuang/junior/PB2panforringed.mirror.gif HTTP/1.0" 200 20222
eger-dl01.agria.hu - - [22/Oct/1998:00:20:51 -0400] "GET
/~tjohnson/pinouts/ HTTP/1.0" 200 26994
```

Distinct from simple web servers, which focus primarily on browsing behavior, web search engines and digital libraries also record data about search and other information seeking behaviors. The following is a sample of a query transaction submitted through the OpenText search engine. It shows the search terms and operations, but also records a good deal of internal cryptic information about how the system operates internally.

```
Mon Sep 28 17:48:42 1998
----- Starting Search -----
Mon Sep 28 17:48:42 1998
{Transaction Begin}
Mon Sep 28 17:48:42 1998
{RankMode Relevance1}
Mon Sep 28 17:48:42 1998
"Bacillus thuringiensis "
Mon Sep 28 17:48:42 1998
P0 = "Bacillus thuringiensis "
Mon Sep 28 17:48:42 1998
R = (*D including (*P0))
Mon Sep 28 17:48:42 1998
R = (((*R rankedby *P0)))
Mon Sep 28 17:48:42 1998
S = (subset.1.10 (*R))
Mon Sep 28 17:48:42 1998
SL0 = (region "OTSummary" within.1 (*S))
Mon Sep 28 17:48:42 1998
(*SL0 within.1 ( subset.1.1 *S ))
Mon Sep 28 17:48:42 1998
(*SL0 within.1 ( subset.2.1 *S ))
Mon Sep 28 17:48:42 1998
{Transaction End}
Mon Sep 28 17:48:42 1998
----- Ending Search -----
```

Digital library systems, most probably for historical reasons, usually implement logs that resemble web log formats or utilize proprietary formats. As an example, below is an annotated sample of a portion of the log of the Greenstone digital library system [23]. Greenstone is a comprehensive, open-source digital library system, which enables logging by setting a specific flag in the configuration file. Each line in the sample user log contains: (a) the IP address of the user's computer; (b) a timestamp in square brackets; (c) the CGI arguments in parentheses; and, (d) the name of the user's browser (Netscape is called "Mozilla").

```
ADMINISTRATION 37
/fast-cgi-bin/niupepalibrary
(a) its-www1.massey.ac.nz
(b) [Thu Dec 07 23:47:00 NZDT 2000]
(c) (a=p, b=0, bcp=, beu=, c=niupepa, cc=, ccp=0, ccs=0, cl=, cm=,
```

```
cq2=, d=, e=, er=, f=0, fc=1, gc=0, gg=text, gt=0, h=, h2=, hl=1,
hp=, il=l, j=, j2=, k=1, ky=, l=en, m=50, n=, n2=, o=20, p=home,
pw=, q=, q2=, r=1, s=0, sp=frameset, t=1, ua=, uan=, ug=,
uma=listusers, umc=, umnpw1=, umnpw2=, umpw=, umug=, umun=, umus=,
un=, us=invalid, v=0, w=w, x=0, z=130.123.128.4-950647871)
(d) "Mozilla/4.08 [en] (Win95; I ;Nav)"
```

The last CGI argument, "z", is an identification code or "cookie" generated by the user's browser: it comprises the user's IP address followed by the timestamp when they first accessed the digital library. The log file usage.txt is placed in the /etc directory in the Greenstone file structure.

Other digital library log formats that we analyzed include those associated with the Dienst protocol (used by the old NCSTRL-Networked Computer Science Technical Reference Library), and the EMERGE, Phronesis, and MARIAN digital library systems.

2.1 Problems with existing DL logs

A careful analysis of the logs of the web and DL systems discussed above reveals a common set of problems. These include:

1. **Disorganization:** Barring a few, most of the system logs were very poorly organized and structured.
2. **Complexity of analysis:** Lack of proper thought in recording the log information makes log analysis a hard problem. Indeed, complex data mining techniques are currently needed to extract some useful information from web and similar types of logs [19, 20].
3. **Incompleteness:** Important information that would be necessary for analysis was omitted from some logs. As an example, most of the logs failed to record the client postal and email address, information that is essential in any user-based study of the system.
4. **Incompatibility:** Each of the systems had their own log formats, making it difficult to use the same tools to analyze logs from different systems for the same kind of study.
5. **Ambiguity:** Many of the log entries and their semantics were not properly and precisely specified in the log format itself, which could lead to ambiguity in analyzing them.
6. **Inflexibility:** The logs recorded a good deal of system specific information which would not be applicable to other systems. This information was recorded in conjunction with other information that was system independent.
7. **Verboseness:** Many of the logs looked just like code dumps used for debugging by the implementers of the system, rather than containing clear and precise information about system usage and behavior.

The above problems were found across the whole set of logs that we analyzed. In the next section, we present our standardized digital library log format design, which attempts to solve many of those problems.

3 The Digital Library Standardized Log Format

As per the previous analysis, current web and digital library logging has a number of problems. Our solution is to propose an XML-based DL standard format which is comprehensive, reflective of the actual DL system behavior, easily readable, precise, flexible to accommodate in varying systems, and succinct enough to be easily implemented.

3.1 DL Log Standard Design

As a first step in creating the DL log format, we collected an extensive, flat set of attributes that we felt were necessary to be recorded in the DL log. The next step was to organize these attributes in a fashion that was logical and structured and could be easily represented and implemented. We chose to produce an XML Schema [21] to formally describe the syntax and semantics of our DL log format. XML provides a standard syntax for the log format; different XML element tags represent different semantic attributes to be registered in the log. As a matter of fact, a similar use of XML to guarantee structural quality of web logs is reported in [24]. XML Schema provides an equivalent to a grammar in XML syntax to specify the structure of the log format. Also, XML log files produced by our tool can be validated against the schema for correctness. Besides that, XML Schema has a rich set of basic types, such as those for numbers, dates, and times, which further contribute to standardization. And finally, the abundance of XML parsers and other related software helps in the construction of analysis tools.

The DL log format had to be reflective of how a generic DL system behaves. We achieve this goal in two ways:

1. By using the 5S digital library theory of Streams, Structures, Spaces, Scenarios and Societies [22] as guidance for how to organize the log structure and define the semantics of the DL components whose behavior would be logged.
   The 5S theory formally defines a standard nomenclature and the semantics of the most common DL components using compositions of mathematical objects. Informally, using the 5S concepts, we summarize that a digital library involves managed *collections* of digital information, accessible over a network, and with associated *services* to support the needs of its communities. Information is manifest in terms of *digital objects*, which contain structured textual or multimedia streams (e.g., images, audio, video). *Metadata* describes different properties of digital objects, and is commonly structured into records. *Collections* and *Catalogs*, i.e., organized sets of metadata records, are stored in persistent, probably distributed *repositories*. In many cases, structures of digital objects and metadata are explicitly represented and explored to improve the quality of services. Basic DL services include indexing, searching, and browsing, and their behaviors are described by means of sets of *scenarios*, which correspond to sequences of user and/or system events and associated actions.

2. By having the notion of a "transaction" as the basic unifying entity of the log format.
   Basically everything that occurs in a DL system could be broken down to the level of a transaction, either as interaction between users and the system or among the system components themselves. Simple examples of a transaction in our format would be a search query submitted by a user, the registering of a new user, or the recording of some system failure. This may be an isolated transaction in a system that does not have the notion of an explicit "session", or it might be a part of a bunch of transactions that define a session. However, most of the current DL log formats, such as CLF, record just one or a few kinds of events or transactions. All or most of the entries in those log files have similar semantics. Our log format is designed to record a number of different kinds of transactions. Examples of distinct transactions are search, browse, session start, etc.

3.2 DL Log format structure

Figure 1 shows the higher-level organization of the DL log format. Each DL log file consists of a number of log entries, each entry representing a type of transaction. Transactions could be categorized as being related to session creation, user registration, user and system events associated with the use of DL services, administration activities, errors, and user-responses. An important and essential feature of the format should be to identify each transaction precisely. To achieve this, we record the timestamp at which it occurred and also associate a unique ID with each transaction. This ID should ideally be monotonically increasing across one server to provide a logical representation of successive transactions. Additionally, in case we're dealing with a non-session based system, we need a way to identify the user. One way to do this is to associate the location (IP address) from which the user is interacting with the system. Each transaction is then associated with a specific statement. A partial XML Schema of the high level organization is shown below.

```
<xsd:complexType name="LogType">
   <xsd:element name="LogEntry" minOccurs="0" maxOccurs="unbounded">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="Transaction"/>
         <xsd:complexType>
           <xsd:attribute name="ID" type="xsd:int/>
         </xsd:complexType>
       </xsd:element>
       <xsd:element name="TimeStamp" type="xsd:dateTime"/>
       <xsd:element name="MachineInfo">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="IPAddress" type="xsd:string"/>
             <xsd:element name="Port" type="xsd:int" minOccurs="0"/>
```

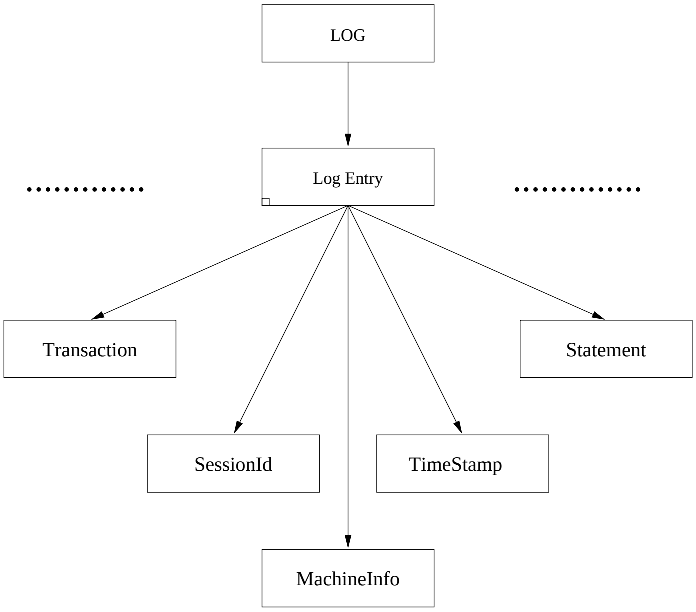

Fig. 1. Top level Hierarchy.

```
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="SessionId" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Statement" type="StatementType"/>
    </xsd:sequence>
  </xsd:element>
</xsd:complexType>
```

There are basically two kinds of statements: 1) those related to specific user and system events associated to DL services; and 2) general statements related to administrative and other general activities. In more detail (Figure 2), the following types of statements are defined:

- **SessionInfo:** In the case of an explicit session based system, the session start and end times, as well as the user's and associated information, need to be recorded. We also assign a globally unique ID to each session. Using this ID, it is very easy to group together all the transactions that occur within this session.
- **Registration info:** In many session-based systems, users have to register themselves with the system when they use it for the first time. They usually have to select a user-ID, password, and possibly provide their identifying and demographic information.

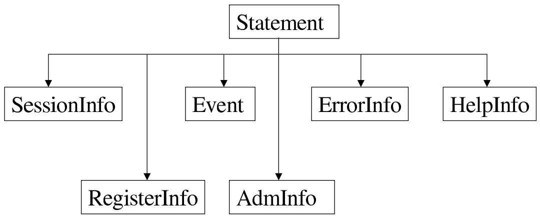

Fig. 2. Decomposition of statement into different types.

- **Administration information:** Most systems record administration activities like system startup, shutdown, backup start, backup end, etc. This transaction type is provided to record such information.
- **Error Information:** This element is related to errors or failures that may occur anytime in the system. Invalid query, document not found, etc., are examples of error and failure information that need to be recorded. If the user forgets to explicitly logout in a session based system the connection time out can be recorded.
- **Help Information:** Some DL systems provide help facilities to aid the user. Use of this feature should be considered to be separate from the other actions described above. Our log format considers this to be another type of transaction. It can be an interesting investigation to find out which kinds of help are frequently used by a user.
- **Event:** We consider this to be the heart of the DL log format. User or system events occur as a result of users performing information seeking activities and using digital library services, or as a system response to those activities. Each event is associated with an action, which encompasses the main operations associated with DL services such as searching, browsing, updating, and recording of system information related to these three operations. Each of those actions is performed over a collection of digital objects or a catalog of metadata. User events also have a status code that is based on the outcome of the action (e.g., success, failure, etc.). Four different kinds of actions are currently defined (Figure 3):
  1. **Search:** Searching is a basic DL service. Different systems implement a number of different query languages and search schemes based on the

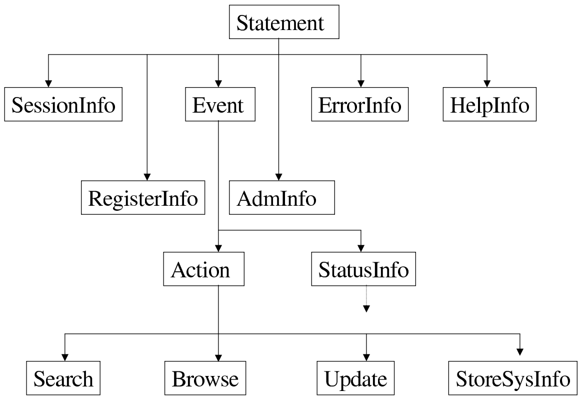

Fig. 3. Decomposition of an event into different types.

underlying retrieval model they use. Two of the common models are boolean and ranked retrieval [5, 4]. Each of these systems also can provide additional features like selection of collection(s), structure related information such as which field the search concerns (author, title, subject, ...), the duration of activities, and some way to indicate whether this search operation is to be performed in the context of a previous, larger search. Systems also can provide options to the users to select how they want to view the results from their queries, including sort option and maximum number of results to be presented. The details of the search element are presented in the portion of the Log Schema below.

```
<xsd:complexType name="SearchType">
  <xsd:sequence>
    <xsd:element name="Collection"
                 type= "xsd:string" minOccurs="0"/>
    <xsd:element name="MetadataCatalog" type= "xsd:string" minOccurs="0"/>
    <xsd:element name="ObjectType">
       <xsd:complexType>
        <xsd:element name="DigitalObject"
                     type="xsd:string" minOccurs="0"/>
        <xsd:element name="MetadataRecord"
                     type="xsd:string" minOccurs="0"/>
        <xsd:element name="HoldingsRecords"
                     type="xsd:string" minOccurs="0"/>
        <xsd:element name="CommunityRecords"
                     type="xsd:string" minOccurs="0"/>
```

```
    </xsd:complexType>
  </element>
  <xsd:element name="SearchBy"
               type= "xsd:string" minOccurs="0"/>
  <xsd:element name="SearchType">
    <xsd:complexType>
     <xsd:element name="persistent"
                   type="xsd:string" minOccurs="0"/>
     <xsd:element name="non-persistent"
                   type="xsd:string" minOccurs="0"/>
    </xsd:complexType>
  <xsd:element>
  <xsd:element name="QueryString"
               type= "xsd:string" minOccurs="0"/>
  <xsd:element name="TimeOut"
               type= "xsd:string" minOccurs="0">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="StartDate" type="xsd:date"/>
        <xsd:element name="EndDate" type="xsd:date"/>
      </xsd:sequence>
    </xsd:complexType>
  </element>
  <xsd:element name="PresentationInfo"
               type= "PresentationInfoType" minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>
```

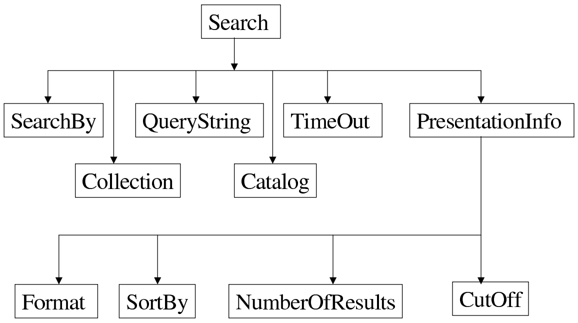

Fig. 4. Search Attributes.

Specific types of objects can be searched including generic digital objects, metadata records, holding records (which keep holding information for real objects in a library system), and community records (non-bibliographic resources that fulfill some information need of a community). SearchBy is used in structured queries and covers specific fields under which the query will be performed. The value of SearchType is set to persistent if the search is to be performed over the result of a previous search. Since query syntax is heavily dependent on the specific DL system and underlying retrieval model, we only record the exact query string used. Log analysts will consider this information in the context of the particular system for their studies. The PresentationInfo includes presentation format (e.g., list, threaded, tabular), which type of sort to apply (e.g., by confidence, by a specific field), number of results, and cut off threshold.

2. **Browse:** Browsing services can be performed by navigation through lists of search results, indexes organized by specific fields, and generic hypertexts. In the browse section we include identifiers of nodes and links navigated, and presentation information.
3. **Update:** Some systems also provide facilities to allow an administrator or user to add, modify or edit some part(s) of collections and/or catalogs resident in a repository.
4. **Store information:** This action allows us to record the data associated with the search and browse actions from the point of view of the system. So, basically actions 1, 2 and 3 above record the user's data, while this action records the system's response data. After any action the system needs to record some information like number of bytes transferred, response time of the action, highest and lowest ranked item, etc.

4 DL Log Tool and its Implementation

The DL XML Log Tool is implemented using generic Java classes and can be used by any digital library or analysis system. There are mainly two classes in the log tool implementation, XMLLogData.java, used for storing data, and XMLLogManager.java, which provides methods to write and read log information according to our DL log format. XMLLogData.java basically provides a structure to hold private data with Set and Get methods to set and get values. For example it has one attribute String SessionInfo for session-based systems and it has SetSessionInfo() and GetSessionInfo() methods to set and get the value of SessionInfo. All the read and write methods are synchronized to avoid conflicts and inconsistences. The most difficult part is how to plug-in the tool into the target system. That should be done by calling specific methods of the XMLLogManager wherever a specific type of transaction occurs. Since this is heavily dependent on the target system architecture and implementation, that should be done by developers or administrators.

First tests were performed on the MARIAN digital library system [12]. MARIAN is a session-based digital library software designed to store, search, and browse large numbers of objects in a distributed environment. MARIAN was originally developed at Virginia Tech in C++ and recently evolved to a pure Java version. MARIAN has a resource management mechanism, which administers and allocates all the system resources such as class managers and searchers. In the MARIAN system, we only have one XMLLogManager Java object in memory, created as an attribute of the ResourceClassManager. Whenever information needs to be logged the client calls the corresponding method of the XMLLogManager instance of the ResourceManager.

4.1 Examples

We have included examples of some log transactions in MARIAN captured from real use of the system. In the examples, we use the Dirline collection, a U.S. National Library of Medicine's online digital library containing location and descriptive information about a wide variety of information resources including organizations and projects concerned with health and biomedicine.

1. Login to the System:

```
<Transaction ID = "3452">
   <SessionId > 987654usr3 </SessionId>
   <SessionInfo>
      <SessionStart> Start </SessionStart>
      <LoginInfo>
         <UserId> mhabib <UserId>
      </LoginInfo>
   </SessionInfo>
   <TimeStamp> 2002-05-31T20:10:55.000-05:00 </TimeStamp>
   <MachineInfo>
      <IPAddress> 128.173.244.56 <IPAddress>
      <Port> 8000 </Port>
   </MachineInfo>
</TransId>
```

2. Query on all Dirline records enties about "low back pain" in any part of the record.

```
<Transaction ID = "3455">
  <SessionId > 987654usr3 </SessionId>
   <TimeStamp> 2002-05-31T20:11:07.000-05:00 </TimeStamp>
   <MachineInfo>
     <IPAddress> 128.173.244.56 <IPAddress>
     <Port> 8000 </Port>
   </MachineInfo>
  <Statement>
    <Event>
      <Action>
        <Search>
         <Collection>Dirline</Collection>
         <ObjectType>CommunityRecord</ObjectType>
```

```
        <SearchBy>SearchByAnyParts</SearchBy>
        <SearchType>NonPersistant</SearchType>
        <QueryString>low back pain</QueryString>
        <TimeFrame>
          <StartTime>2002-05-31T20:11:07.000-05:00</StartTime>
          <EndTime>2002-05-31T20:11:09.000-05:00</EndTime>
        </TimeFrame>
        <PresentationInfo>
          <Format>List</Format>
          <SortBy>ByRank</SortBy>
          <NumberOfResults>217</NumberOfResults>
          <Cutoff>20</Cutoff>
        </PresentationInfo>
      </Search>
    </Action>
    <StatusInfo>successful</StatusInfo>
   </Event>
  </Statement>
</Transaction>
```

3. Browse an item of the ranked list returned as a answer for the previous search.

```
<Transaction ID = "3456">
  <SessionId > 987654usr3 </SessionId>
  <TimeStamp> 2002-05-31T20:11:15.000-05:00 </TimeStamp>
  <MachineInfo>
    <IPAddress> 128.173.244.56 <IPAddress>
    <Port> 8000 </Port>
  </MachineInfo>
  <Statement>
    <Event>
      <Action>
        <Browse>
          <DocID> 5114 </DocID>
          <DocName>University of Washington School of
           Medicine Multidisciplinary Pain Center ( UWPC )
          </DocName>
        </Browse>
     </Action>
    </Event>
  </Statement>
</Transaction>
```

5 Conclusions and Future work

We propose an XML-based digital library log format standard that captures a rich, detailed set of system and user behaviors supported by current digital library services. The format is implemented in a generic log component tool, which can be plugged into any digital library system. Specifications, implementation details, and examples of use within the MARIAN digital library system were described.

Future work will proceed on several fronts. We will be using our log format to allow evaluations of several of our projects, collections and systems, including those in the context of the Networked Digital Library of Theses and Dissertations (NDLTD, www.ndltd.org) and the Computing and Information Technology Interactive Digital Educational Library (CITIDEL, www.citidel.org). Since CITIDEL is a part of the National STEM (Science, technology, engineering, and mathematics) education Digital Library (NSDL, www.nsdl.org), we will advocate use of the log format and tools throughout NSDL. We will test the log tool with other DL systems. A major concern of any comprehensive log format such as ours should be user privacy. We should allow users to choose the level of detail they want the system to log about their activities. Ideally, user information should be logged and maintained at the client side [11] so that users can use that information as they desire, for example, to provide portions of the data to personalization tools in order to get personalization services. We will be investigating extensions in the MARIAN Webgate module to allow such a view.

The current XML format can be very verbose. We will investigate efficient compression techniques to allow scalable analysis of our DL logs. Also, we will consider the application and possible extension of our XML format and tools to support alternative DL architectures, e.g., that of the NCSTRL+ digital library which uses buckets (object-oriented digital objects that contain data, metadata, and the methods for accessing both [9]). Finally, our log proposal needs to be discussed and related to standards and framework activities like OAIS [28].

Acknowledgements

Thanks also are given for the support of NSF through its grants: IIS-9986089, IIS-0002935, IIS-0080748, IIS-0086227, DUE-0121679, DUE-0121741, and DUE-0136690. The first author also is supported by CAPES, process 1702/98-0.

References

1. Borgman, Christine L., Hirsh, Sandra G., and Hiller, John, *Rethinking Online Monitoring Methods for Information Retrieval Systems: From Search Product to Search Process,* Journal of the American Society of Information Science, 47(7) 568-583, 1996.
2. Borgman, Christine L., Personal communication, 1998.
3. Bishop, Ann P., *Digital Libraries and Knowledge Disaggregation: The Use of Journal Article Components,* Proceedings ACM Digital Libraries '98, Pittsburgh, 29-39, 1998
4. Sparck Jones, Karen and Peter Willett, editors, *Readings in Information Retrieval,* San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1997, xv, 589.
5. Frakes, William B. and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms,* Englewood Cliffs, NJ: Prentice-Hall, 1992, viii, 504.

6. Marchionini, Gary, *Information seeking in electronic environments,* Boston: Cambridge University Press, 1995.
7. Marchionini, Gary, *Advanced Interface Designs for the BLS Website: Final Report to the Bureau of Labor Statistics,* http://ils.unc.edu/m̃arch/blsreport98/final_report.html
8. Jones, Steve, Cunningham, Sally Jo, and McNab, Rodger, *Usage Analysis of a Digital Library,* Proceedings ACM Digital Libraries '98, Pittsburgh, 293-294, 1998.
9. Nelson, Michael L., Maly, Kurty, Shen, Stewart N. T., Zubair, Mohammad, *NCSTRL+: Adding Multi-Discipline and Multi-Genre Support to the Dienst Protocol Using Clusters and Buckets,* Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries, IEEE ADL '98, April 22-24, 1998, Santa Barbara, California, USA, 128-136
10. Lynch, Clifford. *Personalization and Recommender Systems in the Larger Context: New Directions and Research Questions (Keynote Speech),* Proceedings of the DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries, Dublin, Ireland, 18-20, June, 2001.
11. Cassel, Lillian N., Wolz, Ursula, *Client Side Personalization,* Proceedings of the DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries, Dublin, Ireland, 18-20, June, 2001
12. Gonçalves, Marcos A., France, Robert K., and Fox, Edward A., *MARIAN: Flexible Interoperability for Federated Digital Libraries,* Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries, Darmstadt, Germany, September 4-9, 173-186, 2001.
13. Barclay, Jean, *Assessing the benefits of learning logs,* Education and Training, Vol.38 No.2, 1996.
14. Riecken, Douglas, *Introduction: personalized views of personalization,* Communications of the ACM, 43(8): 26-28, 2000.
15. Peters, Thomas A., *The history and development of Transaction Log Analysis,* Library Hi Tech, 11(2): 41-66, 1993.
16. Sandore, Beth, *Applying the Results of Transaction Log Analysis,* Library Hi Tech, 11(2): 87-97, 1993.
17. Kaske, Neil K., *Research Methodologies and Transaction Log Analysis: Issues, Questions and a Proposed Model,* Library Hi Tech, 11(2): 79-86, 1993.
18. Gladney, H. M. and Lotspiech, J. B., *Safeguarding digital library contents and users. Assuring convenient security and data quality,* D-Lib Magazine, May 1997.
19. Spiliopoulou, Myra, *Web usage mining for Web site evaluation,* Communications of the ACM, 43(8): 127-134, 2000.
20. Mobasher, Bamshad, Cooley, Robert, Srivastava, Jaideep, *Automatic Personalization Based on Usage Mining,* Communications of the ACM, 43(8): 142-151, 2000.
21. H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn (Eds). "XML Schema Part 1: Structures". W3C Recommendation, May 2001. http://www.w3.org/TR/xmlschema-1/.
22. M. A. Gonçalves, E. A. Fox, L. T. Watson, and N. A. Kipp. Streams, structures, spaces, scenarios and societies (5S): A formal model for digital libraries. Technical Report TR-01-12, Virginia Tech, Blacksburg, VA, 2001.
23. Witten, Ian H., Bainbridge, David, Boddie, Stefan J., *Greenstone: open-source digital library software with end-user collection building,"* Online Information Review, 25(5), 2001
24. Suleman, Hussein, Fox, Edward A., Abrams, Marc, *Building quality into a digital library,.* the Proceedings of the Fifth ACM Conference on Digital Libraries, June 2-7, San Antonio, TX, USA, 228-229, 2001.

25. Davis, Lagoze, Krafft, *Dienst: Building a production technical report server,* Advances in Digital Libraries '95, Springer Verlag, 1995.
26. *Networked Computer Science Technical Reference Library,* http://www.ncstrl.org/
27. *Dienst Protocol,* http://www.cs.cornell.edu/NCSTRL/protocol.html
28. *Open Archival Information System Recommendation,* http://www.ccsds.org/documents/p2/CCSDS-650.0-R-1.pdf
29. *OpenText Search Engine,* http://www.opentext.com