ABSTRACT

This chapter addresses the issues of resource discovery in digital libraries (DLs) and the importance of knowledge organization tools in building DLs. Using the Greenstone digital library (GSDL) software as a case example, we describe a taxonomy generation tool (TGT) prototype, a hierarchical classification of contents module, designed and built to categorize contents within DLs. TGT was developed as a desktop application using Microsoft .NET Framework 2.0 in Visual C# language and object-oriented programming. In TGT, Z39.19 was implemented providing standard guidelines to construct, format, and manage monolingual controlled vocabularies, usage of broader terms, narrower terms and related terms as well as their semantic relationships, and the simple knowledge organization system (SKOS) for vocabulary specification. The XML schema definition was designed to validate against rules developed for the XML
Design and Development of a Taxonomy Generator

taxonomy template, hence, resulting in the generated taxonomy template supporting controlled vocabulary terms as well as allowing users to select the labels for the taxonomy structure. A pilot user study was then conducted to evaluate the usability and usefulness of TGT and the taxonomy template. In this study, we observed four subjects using TGT, followed by a focus group for comments. Initial feedback was positive, indicating the importance of having a taxonomy structure in GSDL. Recommendations for future work include content classification and metadata technologies in TGT.

INTRODUCTION

One of the important contributions of the Web technology to modern information technology development has been the evolvement of digital libraries (DLs) that provide virtual accessibility of information resources to users anytime, from any place where Internet access is available. DLs can be considered as the continuation of the traditional library work in digital form with the support of Internet access. DLs basically involve both primary data and manually created metadata sets. The effort in constructing a DL from scratch is huge since it actually requires an attractive and user-friendly interface with effective content management and powerful search and browsing capabilities.

A solution to this problem was implemented by creating software applications that would help to automate the DL building processes with enhanced facilities to multilingual information retrieval systems, support for interoperability protocols, and effective metadata management for diverse media formats (Witten, McNab, Boddie, & Bainbridge, 2000). During the last decade, several DL building tools have emerged aiming to provide management and distribution processes of DL collections. Some DL software packages may be subject-oriented, institution-oriented, mission-oriented, or used for digital object management. Currently, DSpace, Fedora, EPrint, and Greenstone are the most popular open-source DL software packages available that aim to empower users with capabilities to design, build, and manage digital collections. For example, DSpace offers a platform for digital preservation for an institutional repository system while Fedora provides a service for managing digital objects. EPrint allows open access to digital contents, primarily for institutional repositories and scientific journals. Greenstone is particularly aimed at providing users with easily automated DL building processes.

Despite success in open-source software making DLs more accessible to many users, content management and metadata tagging remain important research challenges in DLs, and, in general, Web portals and Internet resources in facilitating efficient search and discovery of relevant information resources (Yan, 2004). In this chapter, we describe the design and development of a taxonomy generator supporting tagging and classification of digital resources in DLs, focusing on enhancing open-source DL software to provide more efficient search and discovery of information sources.

As a case illustration, we have selected Greenstone (http://www.zdl.org) to implement a taxonomy structure for better resource discovery of the digital collections. In subsequent sections, we give a brief overview of Greenstone, explain the design rationale of the taxonomy generator tool (TGT), and discuss the implementation and initial feedback of TGT. Finally, the chapter concludes with design challenges faced and lessons learned.
OVERVIEW OF GREENSTONE

Greenstone is a software suite designed to build and distribute DL collections for publishing on the Internet or on CD-ROM. It is an open-source application developed under the terms of the general public license (GNU) and is particularly easy to install and use (Witten, 2003). In cooperation with UNESCO and Human Info, Greenstone has helped to support user testing, internationalization, and mount courses (Witten & Bainbridge, 2005). Aligning with the goal of UNESCO for the preservation and distribution of educational, scientific, and cultural information of developing countries, Greenstone came in as an important tool in this context. The core facilities aiming to provide in Greenstone were for designing and construction of the document collections, distributing them on the Web and/or CD-ROM, as well as to providing customizable structure on available metadata, easy-to-use collection-building interface, multilingual support, and multiplatform operation (Witten & Bainbridge, 2005). Although initially focused on helping developing countries, its user base has expanded to 70 countries and the reader’s interface has been translated into 45 languages to-date, with increasing volume of download hits from a steady 4,500 times per month to 6,500 over the last 2 years (Witten & Bainbridge, 2007). Greenstone’s popularity comes from a simple, user-friendly interface providing:

- Essential and efficient basic features: Greenstone is platform independent and multilingual with the capability to handle different types of digital formats from text, image, to video files. End users can access Greenstone through two interfaces, firstly, the reader interface, where end users access the digital collections generally through a standard Web browsers, and secondly, the greenstone’s librarian interface (GLI), where end users can manage the digital resources in five basic activities to gather, enrich, design, create, and format. Tagging the documents, four prespecified metadata sets including Dublin Core (DC) are currently provided with the software and new metadata sets can be created within the librarian interface using Greenstone editor for metadata set (GEMS) (Witten & Bainbridge, 2007). Various search indexes and browsing structures can be configured for each specific collection. Greenstone allows full-text as well as fielded searching across the collections of different index levels and results are usually ranked by relevancy or sorted by a metadata element (Witten et al., 2003). Browsing facility is through the metadata value tagged to each document. Greenstone offers different types of browsing structures (i.e., list, hierarchy, date, etc.) where end users could define for each collection. Unlike other DL building software, Greenstone provides a customizable user interface feature using format string and macro files. Macros are written especially for Greenstone and are loaded at run-time to generate Web pages dynamically (Bainbridge, McKay, & Witten, 2004). Supporting interoperability across different collections and metadata standards, it serves to be compatible with open archives protocol for metadata harvesting (OAI-PMH), Z39.50, and SRW for harvesting and exported to and from metadata encoding and transmission standard (METS) and DSpace collections (Witten & Bainbridge, 2007). For security and administration purposes, Greenstone provides an access control mechanism with a password protection scheme for documents/collections for different levels of end users. At the point of writing, Greenstone is in its Version 3 with an improved, flexible, and extensible design structure. It is now written in Java language and structured as a set of independent modules communicating using extensible markup language (XML).
Design and Development of a Taxonomy Generator

- **Content management:** The survey carried out by Goh, Chua, Khoo, Khoo, Mak, and Ng (2006) shows that Greenstone was able to meet most of the important demands in DLs because of its strong support for end-user functionality, and hence achieving high scores for content management and acquisition compared to other DL building tools such as Fedora, EPrint, and CDSware. The GUI-enabled librarian interface manages the documents in an easy and simplified manner, even for end users without a library science background. Before a collection goes online, it undergoes two processes (Witten & Bainbridge, 2003):
  - **Importing** (import.pl) is a facility to create new collections, add more contents, or delete the collection. Each document and its associated metadata are converted to the XML-compliant Greenstone archive format at this stage of importing. Greenstone allows more than one metadata sets associated to the document collections. Existing metadata sets such as DC are readily available in the system as well as new sets can be created according to user needs (Bainbridge et al., 2004). Metadata in documents and metadata sets are stored as XML files. For each document to be persistent across the collection, an object identifier or OID is calculated and stored as an attribute in the document’s archive file. In order to support extensibility, the task of format conversion and metadata extraction is processed by software modules called plug-ins where each input document is passed through until a suitable plug-in is found. Plug-ins for each specific collection were identified in the collection configuration file (Bainbridge et al., 2004).
  - The building (buildcol.pl) process handles all the indexes and data structures necessary to make the collection work. Greenstone offers three levels of index for each document, that is, document level, section level, and paragraph level, with each described inside the document itself. The building process also compresses the text of the documents for better disk utilization and faster retrieval of the contents. Indexing and compressing techniques are handled by managing gigabytes (MG), which is the core engine in Greenstone that makes use of the GNU database manager program (GDBM) for storing and retrieving the objects by unique keys. To facilitate browsing for available metadata, modules called classifiers are used to create various types of browsers, such as scrollable lists, alphabetic selectors, dates, and arbitrary hierarchies (Bainbridge et al., 2004). Based on classifier types listed in the collection configuration file, the building process initializes each by obtaining the metadata value on which the document is to be classified. With advanced skills on programming, both plug-ins and classifiers can be written to customize functions for new document, metadata formats, and new browsing and document access facilities. Once the building process is finished successfully, the collection is ready to go online and be accessible to the users.

- **Browsing:** What make DLs superior over physical libraries are its better resource discovery capabilities. Greenstone has implemented a browsing system consisting
of classifiers, that is, structures constructed through available metadata in documents. The possible structures in a collection address various levels, for example, section level or document level. During collection building time, documents are grouped into classes according to their metadata while browsing structures are prebuilt and available upon request (McKay, Shukla, Hunt, & Cunningham, 2004). Greenstone supports different kinds of classifiers:

- **List classifier** is the simplest form which presents the sorted documents in a single list structure.
- **Alphabetic classifier** is slightly different than the list structure in which documents are organized into A to Z indexes displayed on top of the alphabetically sorted list.
- **Hierarchical classifier** is more complex than list classifiers with each having an arbitrary number of levels to narrow down the hierarchy, which is specified during metadata tagging (Bainbridge et al., 2004).
- **Date classifier** is like the alphabetic classifier, categorizing the documents by date metadata instead of the A to Z list used (Bainbridge et al., 2004).
- **Phind classifier** creates the classifier not based on metadata values but on a word or the phrases in the documents during collection building time (McKay et al., 2004).

**OUR PROPOSAL**

**Need for Better Classification of Resources in Greenstone**

On the whole, Greenstone provides a new way of collecting and representing the digital contents on the Internet in the form of fully-searchable, metadata-driven DLs, catered with simple, customizable browsing structures for users from novices to experts.

Although the classifier system automates the browsing system, there are weaknesses in the design of the system. McKay et al. (2004) point out two weaknesses to the current classifier system. The first weakness is that because of the ability to switch between searching and browsing, Greenstone treats search and browse features as two standalone systems, displaying search results in a list structure without ranking and sorting facility. Most browsing classifiers built from metadata are not able to search. The second weakness is due to the rigidity of the classifier system in terms of using only static, prebuilt browsing structures presenting a collection in a predetermined format only.

With powerful indexing mechanisms like MG, it offers full-text search to all contents available in the collection. However, in the current classification system of GSDL, the categorization of resources according to subjects is still not supported yet, although the powerful indexing and rich metadata support is already present.

Although Greenstone provides the facilities in multilingual, metadata, and customizable content classifiers for browsing the collection, the current support for browsing lacks a normalized structure for hierarchically organizing the terms. McKay et al. (2004) discuss the current browsing structure of Greenstone and highlight the importance of term suggestion, vocabulary control, and the role of thesaurus in searching and browsing of digital contents. The browsing system in Greenstone is entirely based on the metadata tagged to each document and limits the way it is structured. Additionally, the controlled vocabulary tool for multiple subject access is not managed in the current scenario of Greenstone browsing system. Controlled vocabulary is important to provide the consistent usage of the terminology in the digital resources for better management of indexing and searching capability. Hence, controlled
Design and Development of a Taxonomy Generator

Due to space constraints, we are not able to discuss in detail the steps taken in constructing the taxonomy structure. In summary, five separate processes were involved in designing and developing the taxonomy structure for GSDL:

- **Step 1**: Incorporates controlled vocabularies for taxonomy adapting Z39.19 to determine the terms to be employed in the taxonomy template.
- **Step 2**: Determines the taxonomy scheme to represent various terms suggested in Step 1.
- **Step 3**: Sets the rules to validate against the taxonomy template using the XML schema definition language considered for its powerful support for complex terms.
- **Step 4**: Implements the tool to generate the taxonomy template.

**Figure 1. Implementation processes for TGT**

- **Requirement Specification**
  - Controlled Vocabulary for Taxonomy
  - Terms from SKOS Core Specification
  - XSD for Taxonomy Template

- **System Design**
  - Object-Oriented Concept

- **Implementation**
  - Three-layered Architecture
  - TGT Application (.exe)

- **Methods**
  1. Consult Z39.19
  2. Use XML, SKOS
  3. XML Schema
  4. .NET Framework

**Figure 1. Implementation processes for TGT**

Vocabulary tools such as a taxonomy or thesaurus for richer terms and vocabulary organization are important to incorporate in constructing large knowledge repository using GSDL.

**Aim and Objectives**

Therefore, in this chapter, we describe our research aimed at designing and implementing a taxonomy structure in GSDL to achieve the following objectives:

- To develop a *set of guidelines* for implementing a taxonomy template for GSDL.
- To produce a *taxonomy schema* with the support of controlled vocabulary, semantic relationships, and multiple language support for GSDL.
- To build a *software tool* that would automate the generation of a taxonomy template to be integrated with GSDL.
Design and Development of a Taxonomy Generator

• **Step 5:** Integrates the taxonomy template with GSDL.

Figure 1 shows the overview processes of TGT conceptually grouped onto the software development lifecycle (SDLC) phases.

The TGT prototype was developed using the Visual C#.NET 2.0 and XML technologies and we describe briefly these six areas:

1. **Development platform:** Visual C#.NET is the object-oriented programming language and provides easy setup and installation mechanisms. We used a three-layered architecture to implement this prototype, that is, “forms” for the presentation layer, codes (C# language) for the application logic layer, and XML for the data storage layer. The resultant XML template records each term and their relationships inside the taxonomy hierarchy. There are two solution files included in the project: one for the taxonomy generation application and another one for the setup and deployment project which generates the Windows Installer to distribute the application. The setup project would produce the installer “taxonomy generation tool.msi” file, which contains the application, any dependency files, and application-related information. However, one needs to make assure the prerequisites (e.g., .NET Framework 2.0 to be installed) are well-placed before the installation is made. Otherwise, the process would fail and be back in its preinstallation state.

2. **System requirements:** For optimum application performance of TGT, the machine that the TGT application will be resided in is recommended to have the following system requirements (see Table 1):

3. **Application package:** To install the TGT prototype into the hard drive, the Windows installer file “Taxonomy Generation Tool.msi” has to run. The package will load the step-by-step installation wizard and install the necessary files to the designated location (the default location is ‘C:\Program Files\Taxonomy Generation Tool\Taxonomy Generation Tool’). Once the installation is finished, the application will be accessible from Start Menu or from Desktop Shortcut. From the Start Menu, users are allowed to do two operations: load TGT or read User Guide for assistance in operating TGT.

4. **System design:** Two main functions for this prototype have been developed, namely, creating the new taxonomy template and updating the existing taxonomy template. The help facility includes “User Guidance” and “About” information. Figure 2 shows the overview process flow diagram of TGT.

5. **User interface and features:** The user interface and layout settings are designed as closely related to the existing editor in Greenstone package. The control settings and operation allows multiple language inputs as the term properties are structured to conform to Greenstone editor for metadata set (GEMS). To support the multilingual service, Greenstone uses one XML file named ‘languages.xml’ stored in the folder ‘C:\Program Files\Greenstone\gli\classes\xml’ which contains 139 languages from various countries. Language names and their associated codes, stored in the ‘languages.xml’ file, are retrieved by calling the private method ‘private void prepareComboItems()’ in code behind and populated

### Table 1. Recommended system requirements

<table>
<thead>
<tr>
<th>Operating Systems</th>
<th>Windows XP and later versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>256 MB RAM or higher</td>
</tr>
<tr>
<td>Hard Disk</td>
<td>60 GB with minimum 30 MB of free space</td>
</tr>
<tr>
<td>Processor</td>
<td>1.8 GHz or faster</td>
</tr>
<tr>
<td>Software Requirement</td>
<td>.NET framework 2.0</td>
</tr>
</tbody>
</table>
Design and Development of a Taxonomy Generator

Figure 2. Process flow diagram of TGT

Figure 3. ‘Languages.xml’ from Greenstone digital library

```
<languages>
  <language name="AMHARIC" code="AM">AMHARIC</language>
  <language name="AFAN (Oromo)" code="AF">AMHARIC</language>
  <language name="AFARI" code="AM">AFARI</language>
  <language name="AFAR" code="AM">AFAR</language>
  <language name="AFRIKAANS" code="AF">AFRIKAANS</language>
  <language name="ALBANIAN" code="AL">ALBANIAN</language>
  <language name="ALABAMIAN" code="AL">ALABAMIAN</language>
  <language name="ANDAR" code="AN">ANDAR</language>
  <language name="ANTIGONISH" code="AN">ANTIGONISH</language>
  <language name="ARABIC" code="AR">ARABIC</language>
  <language name="ARABIC-CAUCASIAN" code="AR">ARABIC-CAUCASIAN</language>
  <language name="ARABIC-FRANC” code="AR">ARABIC-FRANC”</language>
  <language name="ARABIC-GERMANIC" code="AR">ARABIC-GERMANIC</language>
  <language name="ARABIC-INDIAN" code="AR">ARABIC-INDIAN</language>
  <language name="ARMENIAN" code="AR">ARMENIAN</language>
  <language name="ASHTARAN" code="AR">ASHTARAN</language>
  <language name="AZERBAIJANI" code="AZ">AZERBAIJANI</language>
  <language name="AZERBAIJANIAN" code="AZ">AZERBAIJANIAN</language>
  <language name="AZERBAIJANIAN-CAUCASIAN" code="AZ">AZERBAIJANIAN-CAUCASIAN</language>
  <language name="AZERBAIJANIAN-FRANC” code="AZ">AZERBAIJANIAN-FRANC”</language>
  <language name="AZERBAIJANIAN-GERMANIC" code="AZ">AZERBAIJANIAN-GERMANIC</language>
  <language name="AZERBAIJANIAN-INDIAN" code="AZ">AZERBAIJANIAN-INDIAN</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <language name="AZERBAIJANIAN-INDIAN-OTHER" code="AZ">AZERBAIJANIAN-INDIAN-OTHER</language>
  <l...
Design and Development of a Taxonomy Generator

inside the Combo box whose selected value will be stored in taxonomy template as ‘xml: lang’ attribute value. The language name and their associated language code used in ‘languages.xml’ file are presented in Figure 3.

Each concept in the taxonomy is seen as an object and stored in memory. An application session-based model is used for temporarily storing the various concept objects in memory using hash table object class ‘ApplicationSession.cs’ (see Figure 4).

The user interface and features are designed to be understandable for users from novices to experts. The simple tab-based incorporation to describe the concepts and their related properties, guiding users step-by-step and with descriptive labels, aims to provide no barriers for users in operating the features provided by the system.

The taxonomy scheme can be created by using a “New” menu item or by simply using the shortcut by pressing Ctrl+N. The “Close” item in the File menu (Figure 5) will unload the active form opened inside the multiple document interface (MDI) while “Exit” menu (Alt+F4) will quit the application without saving anything. Creating the new taxonomy scheme can be logically divided into three parts: describing the taxonomy, describing the concept, and managing the taxonomy tree.

a. **Describing the taxonomy:** Information related to the taxonomy such as name, objective, and scope is recorded before the taxonomy tree is constructed (see Figures 6 and 7).

b. **Describing the concept:** For each label in the taxonomy tree, the associated properties are recorded before it is saved to the taxonomy tree (see Figure 8). The prototype allows for the creation of alternate labels (see Figure 9) and related labels (see Figure 10).

c. **Managing the taxonomy tree:** After the concepts have been described for the taxonomy tree, they are allowed to remove, edit, or save the tree (see Figure 11).

---

**Figure 4. Sample code for ApplicationSession.cs class**

```csharp
public class ApplicationSession : Hashtable
{
    private static ApplicationSession instance = new ApplicationSession();

    private ApplicationSession()
    {
    
    }

    public static ApplicationSession getInstance()
    {
        return instance;
    }

    public override void Add(object key, object value)
    {
        if (this.Contains(key) == true)
        {
            this.Remove(key);
        }
        base.Add(key, value);
    }
}
```
Design and Development of a Taxonomy Generator

Figure 5. File menu

Figure 6. Taxonomy description form

Figure 7. Taxonomy tree description form

Figure 8. Label description tab

Figure 9. Alternate labels tab

Figure 10. Related labels tab
Design and Development of a Taxonomy Generator

Figure 11. Four control buttons for taxonomy tree

Figure 12. Hierarchical tree for fashion taxonomy and its associated nodeID

![Hierarchical tree for fashion taxonomy and its associated nodeID]

Figure 13. Taxonomy template generated for fashion taxonomy for NodeIDs 1-4

```xml
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <skos:Concept rdf:nodeID="1"/>
  <skos:prefLabel xml:lang="en">Fashion Taxonomy</skos:prefLabel>
  <skos:definition xml:lang="en">DDC for Fashion Taxonomy</skos:definition>
  <skos:scopeNote xml:lang="en">This fashion taxonomy is developed using the Costume through time Singapore book developed by National Heritage Board and NLP</skos:scopeNote>
  <skos:narrower>
    <skos:member rdf:NodeID="2"/>
    <skos:member rdf:NodeID="7"/>
  </skos:narrower>
</skos:Concept>

<skos:Concept rdf:nodeID="2"/>
  <skos:prefLabel xml:lang="en">Costume and personal appearance</skos:prefLabel>
  <skos:definition xml:lang="en">Costume and personal appearance regarding to specific age and sexes</skos:definition>
  <skos:altLabel xml:lang="en">Dress</skos:altLabel>
  <skos:altLabel xml:lang="en">Set of clothes</skos:altLabel>
  <skos:altLabel xml:lang="en">Personal style</skos:altLabel>
  <skos:narrower>
    <skos:member rdf:NodeID="3"/>
  </skos:narrower>
</skos:Concept>

<skos:Concept rdf:nodeID="3"/>
  <skos:prefLabel xml:lang="en">Costumes of specific ages and sexes</skos:prefLabel>
  <skos:definition xml:lang="en">Costumes of specific ages and sexes</skos:definition>
  <skos:altLabel xml:lang="en">Dress style relating to specific ages and sexes</skos:altLabel>
  <skos:narrower>
    <skos:member rdf:NodeID="2"/>
  </skos:narrower>
</skos:Concept>
```

83
6. **XML output generated by TGT:** Once the user saves the taxonomy after defining the concepts and their descriptions, a taxonomy template is generated by TGT. The tool was validated by inputting several taxonomy structures, one of which is the sample taxonomy data using the Dewey decimal classification (DDC) of fashion taxonomy. Figure 12, for example, shows the DDC for the fashion taxonomy. Internally, TGT sets each label in the hierarchy a unique id with tag name 'rdf:nodeID'. Figure 13 shows the XML template generated for the hierarchical representation of the fashion taxonomy.

Table 2. Focus group discussion guide

<table>
<thead>
<tr>
<th>Dur</th>
<th>Section</th>
<th>Sub-Section/Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>5mins</td>
<td>1.Introduction</td>
<td>1.1 Greeting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.2 Purpose of the focus group</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Opportunity to provide input about a new software tool for GSDL Software, to assist in generating Taxonomy.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.3 Ground rules</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Role of moderator, Confidentiality of comments, Individual opinions, Speak one at a time and as clearly as possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.4 Brief get acquainted period</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Subjects’ names, positions, institution, Digital Library building experience</td>
</tr>
<tr>
<td>10mins</td>
<td>2.Brief on Objectives of the study</td>
<td>2.1 The areas to be covered in the study</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.2 About taxonomies, thesauri</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.3 Content management</td>
</tr>
<tr>
<td>10mins</td>
<td>3.Demo section</td>
<td>3.1 Give demo presentation &amp; a walkthrough on TGT</td>
</tr>
<tr>
<td>15mins</td>
<td>4.Hands-on trial</td>
<td>4.1 Subject try out the tool</td>
</tr>
<tr>
<td>10mins</td>
<td>5.Initial Evaluation</td>
<td>5.1 Subjects were asked what they like/dislike about the tool and the idea of using XML template as the output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.2 Would it somehow have possibility to improve the resource discovery of Greenstone?</td>
</tr>
<tr>
<td>15mins</td>
<td>6.General Usability</td>
<td>6.1 How user-friendly is this tool?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.2 Likes and dislikes regarding ease of use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.3 Suggestions to improve the tool based on their experience</td>
</tr>
<tr>
<td>20mins</td>
<td>7.Feature Evaluation</td>
<td>7.1 What are the standard features – those they would definitely use?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.2 Are there features they would not use or not require in their own situations?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.3 Are there any features that would be useful provided they were changed in some manner?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.4 Are there any features that would be good in adding as future improvement?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.5 Specific features</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Concept Description Tab</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Taxonomy Tree</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Help capability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Screen layouts</td>
</tr>
<tr>
<td>10mins</td>
<td>8.Consolidation section</td>
<td>Summary of the discussion and comments made by each subjects</td>
</tr>
<tr>
<td>5mins</td>
<td>9.Closing</td>
<td>Any additional comments?</td>
</tr>
</tbody>
</table>
Initial Feedback

In gathering initial feedback on the TGT prototype, we carried out a focus group session with a discussion guide drawn up as shown in Table 2. The objectives of the focus group session were:

- To identify the most important features for users;
- To brainstorm the potential add-on features necessary to fine-tune the tool; and
- To assess and measure the usability and usefulness of the tool.

Positive Feedback

Four subjects (postgraduate and PhD students) at a local university were recruited based on their familiarity with GSDL and the taxonomy concept. Each subject represented various levels of knowledge in understanding and utilizing GSDL and taxonomy.

All the subjects commented that the tool was user-friendly and easy to understand and use. As for using XML template as the output, the subjects thought its flexibility made it easy for them to define their own concepts and make them compatible with Greenstone.

They agreed that TGT followed a good user interface design guideline which could lead the users through a simple and straightforward layout. They seemed to be satisfied with the descriptive labels on top of each frame as it suggested time-saving in capturing the functions and eliminated human errors.

Through the demo section and hands-on experience, the subjects found the core facility useful in allowing for the creation of the taxonomy structure, which includes defining concepts and saving to the tree.

Recommendations

The following recommendations were highlighted by the subjects:

- For better user interface design layout of TGT, intuitive graphics and icons that match the features provided in the system were suggested.
- Subjects pointed out that flexibility in re-locating the concept of the taxonomy was important. Each node in the taxonomy structure should then be allowed to move up and down the hierarchy whenever necessary.
- Subjects showed their satisfaction in using the Help facility in TGT, and commented on the need for a table-of-contents-like hyperlink that goes deeper in the help contents for better “findability” of search item.
- Since TGT followed the design structured in GEMS, the languages selected for the preferred terms and alternate terms should be expressed as the short form in the datagrid. For example, the label description in English means to use the code “en” for the language selected. However, users viewed it as not descriptive and the selected language should be displayed as it is, rather than displaying the code name.
- Subjects recommended that for future extension, the interface itself should be allowed to be express in multiple languages, just like Greenstone does. The labels and help facility could be translated into multiple languages that Greenstone supported.
- Subjects pointed out that in order to improve efficiency in defining related labels for the concept, a subject thesaurus tool can be incorporated for better suggestion of associated terms. This facility could lead to future enhancement of this research work.

CONCLUSION AND RECOMMENDATIONS FOR FURTHER WORK

As browsing and searching are the core functionality for DLs, content management and metadata
Design and Development of a Taxonomy Generator

tagging remain important issues for managing digital repositories and emergent technologies, such as Semantic Web and ontology (Zeng, 2005; Zeng & Chan, 2004). This chapter proposes TGT, a taxonomy generator tool with a hierarchical classification of contents module, designed and built to categorize contents in GSDL. TGT, built upon a common framework for taxonomy construction, is customizable to other DL building environments.

A pilot user study was conducted to evaluate the usability and usefulness of TGT and the taxonomy template. Initial feedback was positive, indicating the importance of having a taxonomy structure in GSDL. Recommendations for future work include content classification and metadata technologies in TGT — a way of addressing data management of resources in Greenstone, focusing specifically on incorporating taxonomy in GSDL and

ACKNOWLEDGMENT

The authors are grateful to David Bainbridge for his inputs on design and implementation of the taxonomy generator, and the subjects for their feedback on the usefulness and usability of the generator.

REFERENCES


KEY TERMS

**Digital Libraries**: They mean different things to different people. The design of digital libraries is, therefore, dependent of the perceptions of the purpose/functionality of digital libraries. To the library science community, the roles of traditional libraries are to: (a) provide access to information in any format that has been evaluated, organized, archived, and preserved; (b) have information professionals that make judgments and interpret us-
users’ needs; and (c) provide services and resources to people (e.g., students, faculty, others, etc.). To the computer science community, digital libraries may refer to a distributed text-based information system, a collection of distributed information services, a distributed space of interlinked information system, or a networked multimedia information system.

**DSpace:** It is jointly implemented by Massachusetts Institute of Technology (MIT) and Hewlett-Packard (HP) laboratories and was released in November 2002. DSpace (see http://www.dspace.org) aims to provide a digital institutional repository system to capture, store, index, preserve, and redistribute an organization’s research data.

**Fedora:** It was originally implemented as a DARPA and NSF funded research project at Cornell University and later funded by the Andrew W. Mellon foundation. Fedora (http://www.fedora-commons.org) offers a service-oriented architecture by providing a powerful digital object model which supports multiple views for digital objects.

**Greenstone:** Greenstone (http://www.greenstone.org) is produced under the New Zealand Digital Library Project, a research project for text compression at University of Waikato. It focuses on personalization and construction of the digital collection from end-user perspectives.

**Metadata:** A set of attributes that describes the content, quality, condition, and other characteristics of a resource.

**Taxonomy:** According to the definition by ANSI/NISO (2005), taxonomy is a collection of controlled vocabulary terms organized into a hierarchical structure with each term having one or more parent/child (broader/narrower) relationships to others. It gives a high level view of contents systematically and provides users a roadmap for discovering knowledge available. Taxonomies can appear as lists, trees, hierarchies, polyhierarchies, matrices, facets, or system maps.

**Usability:** ISO 9241-11 defines usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” Usability of hypertext/Web is commonly measured using established usability dimensions covering categories of usability defects such as screen design, terminology and system information, system capabilities and user control, navigation, and completing tasks.

**Usefulness:** This is debatable. Some make the distinction between usability and usefulness. Although it is impossible to quantify the usefulness of a system, attempts have been made to measure its attainment in reference to system specifications and the extent of coverage of end users’ tasks supported by the system, but not on end user performance testing.