code{4}lib JOURNAL

Issue 12, 2010-12-21 ISSN 1940-5758

Building up a collaborative article database out of Open Source components

Members of a Swiss, Austrian and German network of health care libraries planned to build a collaborative article reference database. Since different libraries were cataloging articles on their own, and many national health care journals can not be found in other repositories (free or commercial) the goal was to merge existing collections and to enable participants to catalog articles on their own. As of November, 2010, the database http://bibnet.org contains 45,000 article references from17 libraries. In this paper we will discuss how the software concept evolved and the problems we encountered during this process.

By Markus Fischer and Stefan Kandera

Introduction

In a meeting of health care libraries in early 2009 in Zurich, Switzerland, we formed a working group with the goal of building up a collaborative article database. Within the network of health care libraries we already had two major database collections which could be used as a starting point for the project: the Rudolfinerhaus in Vienna had a collection of about 23,000 records and the Pro Senectute Library in Zurich had a collection of about 20,000 records. Further growth of the database should be achieved by offering a cataloging tool to the participating libraries.

There is only a small amount of funding for the project: the participating libraries are free to pay a small fee to support the project. From the beginning we strove towards Open Source components, which gave us the possibility to adapt the solutions to our needs.

Practical prerequisites

The working group defined some minimal requirements the end product had to fulfill:

- The reference database should be freely available
- The Software should be manageable for and by libraries
- Open Source solutions are preferred
- The data format should be MARC21 and the rules for cataloging should be AACR2
- The system and the bibliographic data should allow OpenURL requests
- Tracing articles to institutional level is desirable, but not a necessity

The project was built using 5 Open Source components: Vufind as the discovery layer, Doctor-Doc as knowledge database and linkresolver, Koha for cataloging, a set of script utilities to manipulate bibliographic data ("bibnet.org – dedupe utilities") and Drupal as CMS.[1] Indication of availability in Vufind was achieved using the DAIA driver to query Doctor-Doc. Some of these components turned out to be more suitable for our needs than others.

Technical prerequisites

One of the most astonishing facts we were faced with was that MARC21 does not specify separate fields for article level descriptions like volume, issue and page information. While these fields are essential for the most basic functions like building an OpenURL request or to create bibliographic citations, this data is normally stored in MARC21 in the field 773g as freetext:

773 $g 50(2010), no. 3, p. 352-362

The situation we encountered got even even worse, as the MARC data delivered to us came from different institutions with different "rules" on how to enter the information in 773g. It became obvious that it is almost impossible to parse 773g reliably on the fly in a production system out of such diverse data.

We decided to first optimize the MARC structure with additional subfields to hold the citation information, and, in a second step, to parse the data of each institution with an adapted algorithm before importing the MARC21 data into Vufind. This was achieved by analyzing the different contents we found in 773g and by adapting, step by step, the parsing conditions to these situations. The resulting code is included in a GUI based form in the "bibnet.org – dedupe utilities" (see below).

We found no official recommendation to solve this structural MARC problem, except for a discussion paper from the Library of Congress[2] dating from 2003. We chose to use the proposed solution 4.1 from this paper, adding three subfields for volume (773v), issue

(773l) and first page (773q). While this served the basic needs of our project with regard to OpenURL and indicating availability, we now see that even this solution has some shortcomings: for exporting a citation to generate bibliographies you also need the last page. We may address this in future improvements.

Since many Open Source library systems like Koha or Vufind are built around MARC21, it is inevitable that these systems, as article databases, have the same shortcomings as the MARC21 format itself: missing fields like volume, issue and first page in every record based function of the system. Cataloging articles does not seem to be the traditional core business of libraries.

Machine readable accessibility to the data stored in the freetext field 773g turned out to be essential for our project. The solution we currently run is the following:

$g 50(2010), no. 3, p. 352-362 **$l** 3 **$q** 352 **$v** 50

For the reasons explained above, rethinking these structural MARC21 problems would be an important step in making bibliographical article data usable independently of its holding data, and finally, suitable for search systems supporting OpenURL.

Vufind

http://sourceforge.net/projects/vufind/files

Vufind is a discovery layer for libraries. The System is based on a SOLR index and sets new benchmarks in discovering library collections. Vufind is developed by Villanova University. We found that Vufind is easily expandable due to a very clean and modular design. We made a simple default installation of Vufind and added the following changes:

- Additional fields for volume, issue, startpage in every aspect of the view (overview, details, favorites, export record, cite this...)
- A location chooser to manually override the indication, by IP address, of availability of articles for an institution
- Changed the DAIA-Driver to send availability requests over OpenURL and add the IP of the requesting client
- Speed up the availability requests by parallelizing the sequential requests, using CURL

Location chooser

The institutions listed in the location chooser, added in our installation below the existing language chooser, can be defined in Vufind's config.ini as a key value pair together with their IPs. The default indication of availability is by IP address of the requesting client, but this can be overridden by selecting another institution in the location chooser. The code for the location chooser is placed in web/index.php and simply sets a cookie with the IP of the desired institution to view its availability:

view plain copy to clipboard print ?

```
01. // Setup Locator
02. $ipRequest = $_SERVER['REMOTE_ADDR'];
03. if (isset($_POST['myloc'])) {
04.     $location = $_POST['myloc'];
05.     setcookie('location', $location, null, '/');
06. } else {
07.     $location = (isset($_COOKIE['location'])) ? $_COOKIE['location'] :$ipRequest;
08. }
09. // Make sure the location code is valid. Reset to default if not:
10. $validLocations = array_keys($configArray['Locations']);
11. if (!in_array($location, $validLocations)) {
12.     $location = $configArray['Site']['location'];
13. }
14. $interface->setLocation($location);
```

There were other small changes involved. To make the location chooser appear as a select button in the GUI you need to add it in layout.tpl:

view plain copy to clipboard print ?

```
01. {if $showBreadcrumbs}
02.   <form method="post" name="locForm" action="">
03.       <select name="myloc" onChange="document.locForm.submit();">
04.         {foreach from=$allLocs key=locCode item=locName}
05.           <option value="{$locCode}"{if $userLoc == $locCode} selected{/if}>{translate text=$locName}</option>
06.         {/foreach}
07.       </select>
```

```
    <noscript><input type="submit" value="{translate text="Set"}" /></noscript>
  </form>
{/if}
```

In web/sys/Interface.php you need to add

view plain copy to clipboard print ?

```
function getLocation()
{
    return $this->location;
}

function setLocation($location)
{
    global $configArray;
    $this->location = $location;
    $this->assign('userLoc', $location);
    $this->assign('allLocs', $configArray['Locations']);
}
```

CURL

CURL is a library to facilitate getting and sending files or requests using a URL syntax. Since PHP does not include threading in its core, CURL might be used to create a multi-threaded type situation for sending and receiving requests over the web. To speed up the rather slow availability drivers in Vufind we installed CURL on the server and tweaked the DAIA driver to send “multi-threaded” OpenURL requests. The base code for this is from: http://www.phpied.com/simultaneuos-http-requests-in-php-with-curl/ by Stoyan Stefanov, which we adapted to our needs:

view plain copy to clipboard print ?

```
private function multiRequest($data, $options = array()) {

// array of curl handles
$curly = array();
// data to be returned
$result = array();

// multi handle
$mh = curl_multi_init();

// loop through $data and create curl handles
// then add them to the multi-handle
foreach ($data as $id => $d) {

  $curly[$id] = curl_init();

  $url = (is_array($d) && !empty($d['url'])) ? $d['url'] : $d;
  curl_setopt($curly[$id], CURLOPT_URL,            $url);
  curl_setopt($curly[$id], CURLOPT_HEADER,         0);
  curl_setopt($curly[$id], CURLOPT_RETURNTRANSFER, 1);

  // post
  if (is_array($d)) {
    if (!empty($d['post'])) {
      curl_setopt($curly[$id], CURLOPT_POST,       1);
      curl_setopt($curly[$id], CURLOPT_POSTFIELDS, $d['post']);
    }
  }

  // are there any extra options?
  if (!empty($options)) {
    curl_setopt_array($curly[$id], $options);
  }

```

```
35.   curl_multi_add_handle($mh, $curly[$id]);
36. }
37.
38. // execute the handles
39. $running = null;
40. do {
41.   curl_multi_exec($mh, $running);
42.   usleep(25000); // important to reduce unnecessary CPU-load!
43. } while($running > 0);
44.
45. // get content and remove handles
46. foreach($curly as $id => $c) {
47.   $xmlString =
48.   $result[$id] = curl_multi_getcontent($c);
49.   curl_multi_remove_handle($mh, $c);
50. }
51.
52. // all done
53. curl_multi_close($mh);
54.
55. return $result;
56. }
```

DAIA

http://www.gbv.de/wikis/cls/Document_Availability_Information_API_%28DAIA%29

DAIA (Document Availability Information API) is a data model to encode availability information for documents. The protocol was developed by the German consortias GBV[3], HeBIS[4] and the Beluga[5] project. DAIA is a simple and lightweight model to query and return the availability of a given document. It works on the assumption that you have a unique identifier / ID associated with the document.

Articles can hardly be traced for availability using a unique record ID if they are not mapped to holdings. Identifiers like DOIs or PMIDs are often not available for the kind of literature we are indexing. Additionally we do not come from a consortial situation with one shared ILS. Typically there is no one-to-one relationship where a record has a unique ID for one article related to holdings information. Therefore, we were forced to obtain availability information by using the bibliographic information of ISSN, year, volume and issue, from the MARC 773g field.

To make DAIA work for our purposes, we send OpenURL requests to our DAIA server[6] and get back DAIA-XML answers. An institution's IP address can also be appended as a parameter to the OpenURL request, to get the availability of a record for that institution. This allows Vufind to distinguish between availability by IP address on the records overview and the general availability for all institutions in the details of a record.

Doctor-Doc

http://sourceforge.net/projects/doctor-doc/

Doctor-Doc is primarily a tracking tool for ILL. It may also be used as a link resolver for online journals in connection with the German EZB[7]. Journal print holdings can be uploaded[8] to Doctor-Doc, to be indicated in the link resolver and searched down to issue level through a DAIA interface[9] sending OpenURL requests to Doctor-Doc.

There is one big challenge when trying to establish availability information by using the ISSN as your main identifier: a journal often does not have one, but several ISSNs (like E-ISSN, P-ISSN and an ISSN for the CD-Rom edition). Often a journal gets a new ISSN when there is a small change in the title or when a journal changes its publisher. ISSN.org recently introduced the L-ISSN (linking ISSN) that should be used in OpenURL situations to create a more reliable linking situation. But many libraries and data providers still do not use L-ISSN.

Doctor-Doc takes care of this messy ISSN situation by internally mapping an ISSN to any related ISSN. So a client may do an OpenURL request with the most recent version of an ISSN and still finds all holdings that use the old version of the ISSN. It turns out that for our situation, after having resolved the issue with multiple ISSN numbers, the ISSN is a reliable identifier. The ISSN data to achieve this comes primarily from the ISSN-to-ISSN-L[10] table of all ISSNs assigned by ISSN.org, which has been freely available for download on their website.

Koha

http://koha-community.org/download-koha/

Since Vufind is not a cataloging system but a highly optimized search interface using an index instead of a database, we had to find another system able to produce MARC21 data for those libraries without a capable ILS. We didn't find any small system that would be appropriate for our needs, so we chose Koha. Koha is an Integrated Library System (ILS or ILMS). We are using the cataloging editor and the MARC21 framework for about 400 Journal templates and the authority file system. We run version 3.0.6.

While Koha is a very large and powerful system, it is not easy to customize due to a rather complex design. We had to fix various Java-Script bugs in connection with IE on the cataloging interface, which failed to open new windows, due to blank space in the wrong place in the JS function. We needed to ensure that the cataloging tool works with Browsers down to IE 7.

There is an ongoing bug in the MARC 008 field in Koha which scrambles the content of the field. We have been forced to correct the data before importing into Vufind, by using a custom de-dupe and data control tool (see "bibnet.org – dedupe utilities" below).

A nice feature in Koha is that cataloging templates can be generated for different materials. We created MARC21 templates for about 400 journals with journal title and ISSN, mandatory, and repeatable or default values already defined. Based on our experience with both commercial and Open Source ILS, we realized that there is a high potential for optimizing cataloging articles. Catalog records for articles contains a lot of repeated information which should be system generated rather than user entered:

Koha specific

- The leader and the system-ID is system generated but needs an additional click each.
- The system-ID should not be editable once generated, to avoid loss of record identity.
- Copying an existing record should produce a different system-ID.
- The code for the cataloging agency should be prefilled and not be editable.

General

- The code for the publication place in 008 should be automatically prefilled for the selected journal.
- The language code in 008 should be automatically generated for the selected journal.
- Publication date in 008 could be generated with the actual year.
- The volume in 773v could be calculated upon the frequency of the publication.
- 773g could be composed automatically from 260c, 773v, 773l and 773q.
- During the cataloging session the last entered issue information should be maintained for the next article to avoid reentering of 260c, 773v, 773l and 008.

We are currently evaluating if we should develop a custom application for the specialized task of cataloging articles.

bibnet.org – dedupe utilities

http://sourceforge.net/projects/bibnet/

A major task in the project is de-duping MARC21 data coming from different sources. Vufind and Koha both provide de-duplication mechanisms. However, we found neither solution was flexible enough for our situation. Both approaches de-dupe either too much or too little. De-duplication is dependent on the structure of a given record. A record with full citation information may be de-duped more selectively than a record containing only a title and a year. So we developed a custom de-duping application, available as Open Source in a raw but functional beta stage.

Basically this de-dupe utility performs data control and de-dupes using different approaches depending on the machine readable information (ISSN, volume, issue, startpage) present in a record. Possible duplicate matches are further checked and identified by using a fuzzy search on the title. The fuzzy search is accomplished by calculating the Damerau-Levenshtein distance after string normalization. It is essential to normalize the strings being compared before using this algorithm.

Librarians tend to index records with different punctuation, using different case sensitive characters and so on. We normalize the article title to lower case and remove all spaces and non-alpha-numeric characters before calculating the Damerau-Levenshtein distance. We found that using this approach results in highly reliable de-duplication for our needs.

Drupal

http://drupal.org/

For a CMS we chose Drupal. Drupal serves for coordination purposes, although any other CMS could do as well:

- we create the internet presence of the project.
- we provide direct links from Drupal to the cataloging templates of Koha.
- we list the journals assigned to libraries to be cataloged.
- An internal ticketing system (Drupal modul “Support 6.x-1.3”) helps us coordinate different tasks (basically, to correct bugs and report new feature requests).

Conclusions

http://bibnet.org/

The features required by bibnet.org could not be developed in a single architecture without significant investments. As a consequence we chose a modular architecture with different software components. Not all components of bibnet.org work together without manual intervention at the moment. In particular, the export, de-dupe and import processes need further automation. We are confident we can achieve this after creating a more customized and automated cataloging situation with less possibility of cataloging errors.

The concept of bibnet.org can be adapted to any pool of libraries willing to work together. We try to convey self-administration to any participating library with as little technical interference as possible. Each library is invited to work with (or improve) each component. The code (and therefore the system itself) is entirely open:

- CMS (Drupal) for allocating libraries and journals, to coordinate cataloging and for documentation purposes
- ILS (Koha) for cataloging
- “bibnet.org – dedupe utilities” to prepare records for import
- OPAC (Vufind) as discovery layer
- Doctor Doc as an ISSN and holdings knowledge base to denote availability (DAIA) and as a link resolver using the services of the German EZB

The most challenging part is to change MARC21 to be able to expose machine readable article reference data. As a consequence all incoming data from external resources have to be parsed by our “bibnet.org – dedupe and script utilities”. We think the gain, especially in regard to OpenURL functionality, is worth the effort.

Koha turned out to be less suitable for our needs because of incompatible behavior with IE (due to inappropriate use of Java-Script in the cataloging interface) and because of some other bugs like the 008 bug we mentioned earlier. In general we found that for cataloging articles the interface of Koha is not optimized to the point it could be. As powerful as Koha may be as a general full featured integrated library system, it seems less apt for our specialized needs.

bibnet.org shows the potential libraries can have if they start to cooperate: the software components available for the library world today do allow building aggregate systems with a reasonable effort. We can recommend in particular Vufind, which is easy to install and to adapt. Vufind provides a powerful discovery system for a multi-institutional environment.

Future plans

Cataloging

Develop a streamlined and optimized cataloging solution for bibnet.org to produce adapted MARC21 data. This could be achieved by expanding the existing code base of “bibnet.org – dedupe utilities”. While building a general ILS is a huge and complex task, we believe that creating a specialized cataloging system for article references is rather easy to achieve.

OAI

Vufind recently integrated a functional OAI harvester and import function. We plan to integrate OAI connectors to import external data and to further expand the existing database. Cairn.info[11], Heclinet[12] and CCMed[13] could be valuable targets and partners for bibnet.org.

Alerts

Many publishers of the journals we index do not provide alert functionality for their publications. We are aiming to create such functionality within Vufind or within the eventually expanded code base of “bibnet.org – dedupe utilities”. Vufind recently added the possibility of tracking the index date of a record by providing a persistent entry in a database. This is a precondition to develop alert functionality for Vufind, since the SOLR index is not an optimal place to permanently store this kind of information.

Notes

[1]We leave out the evaluation process here, but other Open Source Software solutions were tested as well.

[2]http://www.loc.gov/marc/marbi/2003/2003-dp01.html

[3]GBV -Gemeinsamer Bibliotheksverbund: http://www.gbv.de

[4]HeBIS -Hessisches Bibliotheks- und Informationssystem: http://www.hebis.de/

[5]Beluga: http://beluga.sub.uni-hamburg.de/

[6]DAIA server address of Doctor-Doc: http://www.doctor-doc.com/version1.0/daia.do

[7]The “Elektronische Zeitschriftenbibliothek Regensburg” is a freely accessible knowledge database and A-Z list for scientific online journals. http://rzblx1.uni-regensburg.de/ezeit/

[8]Information about the upload process for print holdings to Doctor-Doc: http://sourceforge.net/apps/mediawiki/doctor-doc/index.php?title=Help:Contents#Upload_print_holdings

[9]Description of the DAIA interface in Doctor-Doc: http://sourceforge.net/apps/mediawiki/doctor-doc/index.php?title=Help:Contents#Print_holdings_availability

[10]http://www.issn.org/2-24117-Download-the-ISSN-ISSN-L-table.php

[11]Cairn.info databse of human and social science journals: http://www.cairn.info/

[12]HECLINET Health Care Literature Information Network (stopped in 2000): http://www.dimdi.de/static/de/db/dbinfo/hn69.htm

[13]CCMed: http://www.zbmed.de/ccmed.html

About the Authors

Markus Fischer is head of hospital libraries “Solothurner Spitäler AG”. He is the main developer of doctor-doc.com and swissconsortium.ch (http://www.so-h.ch). He can be reached at: markus.fischer@spital.so.ch

Stefan Kandera is librarian at “Bibliothek und Dokumentation Pro Senectute Schweiz” (http://bibnet.org/?q=taxonomy/term/2). He graduated in Philosophy at the University of Constance (Germany) and has a Master of Advanced Studies in Information Science (HTW Chur, Switzerland). He can be reached at: stefan.kandera@pro-senectute.ch