QUEENS COLLEGE OF THE CITY UNIVERSITY OF NEW YORK

ADVANTAGES AND CHALLENGES OF OPEN SOURCE WEB PUBLISHING: A CASE STUDY

A RESEARCH PROJECT SUBMITTED TO DR. COLLEEN COOL OF THE GRADUATE SCHOOL OF LIBRARY AND INFORMATION SCIENCE AS A REQUIREMENT FOR COMPLETION OF THE DEGREE OF MASTER OF LIBRARY SCIENCE

BY
SCOTT VOTH
scott.voth@verizon.net

FLUSHING, QUEENS
MAY, 2010

Abstract

Can open source Web publishing software make a digital collection more accessible, interesting, and easier to manage? This case study took a convenience sampling of 630 vintage postcards from the *Waterways of New York* collection, hosted by CONTENTdm (http://cdm128401.cdmhost.com/cdm4/browse.php?CISOROOT=%2Fqcgslis_f06), and migrated the images and metadata to Omeka, an open source web publishing platform. The experience of setting up and customizing the new site (http://pcnyw.net/omeka/) were carefully recorded and analyzed. One design goal - to display each postcard as an information object, with a front and a back image - complicated the project, and extensive modifications to the Omeka theme were required. Importing the postcards was arduous, and Omeka enhancements suggested. Quantitative comparisons between CONTENTdm and Omeka were performed, including an access point count, and an evaluation of the two platforms' advanced search functionality. Omeka's plug-ins were evaluated, including its Google Map mash-up and its interpretative exhibit functionality. Admin panels were investigated, and the "Add" and "Edit" functionality reviewed. A slideshow was developed using an external open source package, and Omeka's extensibility tested out. Omeka has some very powerful features that make it attractive to museums, libraries and archives, but it is not yet mature. Its success depends upon the dedication of its open source community, and that dedication appears to be growing.

Table of Contents

Abstract……..2

Chapter I – The Problem…….6

Introduction……....6
Background of the Problem…….4
Problem Statement……..8
Why Omeka? 9
Objectives ……...9
Research Questions…….7

Chapter II: Literature Review……...….11

Background of the Open Source Movement……..11
Characteristics of Open Source……...…12
The Open Source Community – Demographics and Motivations…….13
Group Forums and their Role in Defect Tracking and Resolution……....15
Evaluating and Implementing Open Source Solutions……..16
Digital libraries and Open Source…….….17
Literature Review Summary…….….18

Chapter III: Methodology…….…….…20

Overview and restatement of problem…….…….….20
Units of Analysis……20
Unit 1 - Setup and Customization…….…….……21
Unit 2- Importing Content…….……21
Unit 3 – Content Presentation…….…….……..23
Unit 4 – Administration…….....24
Conclusion…….24

Chapter IV – Findings……...25

Background and Timeline…….…….25
Setup & Customization……..……25
Initial Setup……...…….25
Plug-ins …….……27
Theme Considerations……...29
Support and Documentation……..……34
Conclusions about Setup and Customization…….34
Importing Content…….35
Reformatting the data exported from CONTENTdm……35
Image preparation…….…….35
Importing the postcards…….……36

Measuring import time…….…….37
Conclusions about Importing Content……..……39
Content Presentation…….……40
Access Point Comparison…….……40
Retrieval Comparison…….…….……..40
Exhibit Building.……42
Browse by Tag……..…..…….44
Slideshow……..44
Additional Context Pages…….……45
Find on Map…….……..45
Conclusions about Content Presentation…….……..46
Administration……...47
Adding and Editing Items……47
Rights and Permissions……..48
Conclusions about Administration……....49
Summary of Findings……49

Chapter V – Conclusions……..…….51

The Open Source Model and Omeka…….……..…….51
Lessons Learned…….……..…….53
Additional Research Opportunities……..…….……54
Acknowledgements……..…….55

Appendix 1 – Final Evaluation Criteria…….56

References…….57

Subject Index…….61

Table of Figures

Figure 1 - Plug-in Installation and Configuration Screen 27
Figure 2 - Navigation and Welcome Page 31
Figure 3 - Displaying a selected postcard showing both front and back sides 32
Figure 4 - LightBox Image Handling 33
Figure 5 - CSVImport Adminstration Panel 38
Figure 6 - Time it took to import the postcards 39
Figure 7 - CONTENTdm and Omeka Search Comparison 42
Figure 8 - Find on a Map 47
Figure 9 - Omeka Admin Panel for Editing Items 49

Advantages and Challenges of Open Source Web Publishing: A Case Study

Chapter I – The Problem

Introduction

Open source Web publishing software has yet to eclipse the proprietary market, but its prevalence certainly warrants an investigative study. Free to download and customize, this software is developed by a community of programmers who believe that quality and collaboration are inherently linked. What are the advantages and challenges of using open source to publish a digital collection?

This case study used a convenience sampling of 586 postcards from *Waterways of New York (Waterways),* a digital collection currently housed on the proprietary content management system CONTENTdm. The postcards were migrated it to a new, open source platform called Omeka to explore how new access points can be developed to take advantage of the collection's rich metadata and make the collection easier to search and enjoy. Experiments were conducted using Omeka's plug-ins to craft interpretative exhibits and implement Web 2.0 tools which allow users to find postcards on a Google Map, watch a slideshow, and search by tags.

Background of the Problem

Waterways is an on-going, instructional project that involves scanning and cataloging vintage postcards from the Erie Canal, the Hudson River, New York Harbor, the Champlain Canal and Long Island Sound. All the postcards were originally created between 1898 and 1923, and their copyright is now in the public domain. Each postcard in the collection has been carefully digitized and its metadata collected by students of the

Graduate School of Library and Information Science at Queens College (CUNY) in a class called *Introduction to Digital Imaging*. Funding to publish the collection was provided by Metropolitan New York Library Council, and *Waterways* is currently part of the organization's larger collection called *Digital Metro New York*, which includes eighteen other digital collections from colleges and libraries in the New York metropolitan area. *Digital Metro New York* is partly funded by the New York State Regional Bibliographic Databases Program (Digital Collections, n.d.), and is powered by CONTENTdm, a proprietary web publishing engine supported and marketed by OCLC (Online Computer Library Center).

Hosting the collection under these circumstances has proved troublesome for the professors who teach *Introduction to Digital Imaging* at Queens College. According to Professor Perry (2010), CONTENTdm is often unavailable, and files often "go missing" (Perry, 2010, para. 2). In 2009, the site was unavailable for over a month for software upgrades. This was at a point when the class was anxiously waiting to load their postcards.

As currently published on CONTENTdm, the *Waterways* collection is drab and unappealing. The collection is presented with sixty-three consecutive pages, each with five rows of four thumbnail shots of the postcards. The user may click on each card, but there is no correlation between front side and backside of each card. As Bullen (2008) notes, digital repositories are very good at storing and accessing images, but most have no "coherent mechanism for placing the items in context" (p. 31). Bullen's solution is to keep his digital collection inside CONTENTdm and add a CMS layer (he has

experimented with both WordPress and MediaWiki) to add textual content to his collection.

But for the *Waterways* collection, CONTENTdm appears to be the wrong publishing tool for two reasons, both of which relate to its proprietary nature. The first is collection management. As manager of the Waterways collection since 2005, Perry (2010) cannot remember "a single semester in which serious problems with the system software did not arise." (para. 3). As previously mentioned, missing files, unscheduled upgrades, limited support, and unexplained outages all make managing a basic CONTENTdm collection inconvenient and difficult. As Perry (2010) notes, CONTENTdm is a for-profit system, and customization and improvements are available, "but at a price." Libraries with small budgets have little chance to upgrade to the product's latest toolsets, and are at the mercy of outages, updates, and poor customer support. The second, and perhaps most important reason CONTENTdm doesn't work for the collection pertains to aesthetics. The final product is unappealing and monotonous. It has few access points. There is no opportunity to add context to the collection so that it may be viewed in perspective. Perry states that every CONTENTdm collection tends to look the same, reducing its "'wow' factor' … and the ability to optimize on a branding opportunity" (para. 1). She goes on to note how many other platforms have become more innovative by using Web 2.0 tools, and she is anxious to consider alternatives (para. 3).

Problem Statement

Queens College does not have funding to mount the *Waterways* collection on a proprietary host and receive premium services and customization. Instead it is forced to share a base CONTENTdm system with eighteen other collections, and is subjected to

poor service. CONTENTdm provides few access points and poor retrieval functionality. The user interface is drab and unappealing. The collection's potential is unrealized. Can an open source alternative make the collection easier to manage and bring it to life?

Why Omeka?

Omeka is an open-source software platform developed by the Center for History and New Media at George Mason University. As a "web-based publishing platform for scholars, librarians, archivists, museum professionals, educators, and cultural enthusiasts" (Omeka About Page, 2009, para. 1), Omeka closely resembles WordPress, with its plug-ins and themes. Founded in 2007 and first released in 2008, Omeka is not yet mature as a platform, and has only a dozen plug-ins and six themes. It is becoming widely used by museums, archives and libraries. It was chosen because it is representative of the various open source web publishing options currently available and also because it may reveal much about the challenges libraries face when they choose such an approach to publish their digital collections.

Objectives

Clearly the results of a single case study are insufficient grounds to verify the merits open source software, and this certainly is not the intention of this study. However, a case study can provide valuable insight for those set to embark on similar tasks, using similar tools. This study created an initial evaluation framework that listed items to evaluate, and used it to record personal observations as the migration progressed. Unexpected results were added to these items, and a list of lessons learned and pitfalls to avoid was compiled.

Research Questions

The following research questions were addressed. First, how can open source Web publishing improve upon the proprietary software model? Second, what are the challenges of using open source when publishing a digital collection? And third, what are the lessons learned from migrating such a collection to an open source platform?

Chapter II: Literature Review

Background of the Open Source Movement

Three acronyms, OSS (Open Source Software), FSM (Free Software Movement) and FOSS (Free Open Source Software), and the term "software libre," have been used to describe the open source software movement, which is widely believed to have started in 1983 when Richard Stallman founded the GNU Project and formulated and implemented his "GNU Public License" that defined the rights of users to review source code, make changes to it, and redistribute it, either in the original or modified versions, at no cost (von Krogh & von Hippel, 2003, p. 3). In 1985, Stallman created the Free Software Foundation, and he is still the acting president (Free Software Foundation, 2009) of this "copy-left" movement set up to protect the rights of software users. (Stallman, 1983, para. 1-4).

But as von Krogh & von Hippel (2003) note, the OSS movement had earlier roots. The "communal hacker culture" of the 1960s and 1970s, especially prevalent around MIT and in ARPANET, was based on the free exchange of software. Stallman was a part of this culture, and watched as the proprietary software market developed, and condemned it as "morally" wrong (p. 4), using the term "software hoarding." (Williams, 2002).

In contrast to FSF, the Open Source Initiative (OSI) was founded in 1998, and differs from FSF in that it is less political and confrontational, and focuses instead on the superiority of open source software. (History of the OSI, para. 16-18). While the two groups have similar goals and work together on many projects, OSI focuses on the "practical benefits" of the various licensing agreements that have subsequently evolved (von Krogh & von Hippel, 2003, p. 4). O'Reilly (1999) notes how this has lead to many

different offshoots, including some that might be “anathema” to members of FSF, such as the “gated open-source community” model, which licenses out software rights only to its large (paying) user base, and reaps benefits from it users’ modifications, documentation and defect resolution. Another model is to freely distribute source code, and allow modifications and redistribution for noncommercial use, but when used commercially, a different license is required. This is the model which Sun’s Java project follows (p. 36).

Characteristics of Open Source

In its website, the OSI lists ten criteria to which software must comply in order to be considered open source. These are: (1) “Free Redistribution” – no restrictions on the sale or sharing can be stipulated in the license; (2) “Source Code” – the source code must be included so that everyone can read and modify it; (3) “Derived Works” – any modifications to the source code can be distributed under the same terms of the license; (4) “Integrity of The Author's Source Code” – a license may be constructed to restrict modifications only if it “allows the distribution of ‘patch files’ with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code.” Also, a license may require that the modified versions be named differently from the original; (5) “No Discrimination Against Persons or Groups” – license cannot discriminate; (6) “No Discrimination Against Fields of Endeavor” – license cannot restrict the use of the software to a certain field; (7) “Distribution of License” – additional licenses are not necessary when the program is redistributed; (8) “License Must Not Be Specific to a Product” – if the program is part of a larger distribution, the license should guarantee the right to extract and use as desired; (9) “License Must Not Restrict Other Software” – license cannot

"insist that all other programs distributed on the same medium must be open-source software"; and (10) "License Must Be Technology-Neutral" – license cannot require that the software be used with a specific technology (The Open Source Definition, n.d., para. 1-10).

As O'Reilly (1999) notes, giving programmers access to source code and the right to create derivations allows them "to help themselves, and encourages natural product evolution as well as preplanned product design." He sees this methodology as clearly superior when creating customized software (p. 34). Coders are encouraged to look under the hood, find bugs and fix them. They are invited to add on to the code, create extensions or plug-ins, or take the code base and re-tool it into something new.

OSS is thoroughly dependent upon a development community. The "key determinant of a project's success" is not the product itself, but rather the support of the user community. The project must attract a "passionate core of users" who are anxious to expand and extend it, and who are willing to devote their time and energy, often with little or no financial compensation. The "glue" that holds a development community together is cooperation rather than money (O'Reilly, 1999, p.36-37).

The Open Source Community – Demographics and Motivations

One question which von Krogh & von Hippel (2003) try to answer is why so many excellent developers contribute so freely to open source development. Their introductory article to *Research Policy*'s special edition on Open Source compiles a succinct review of four articles which concern the motivations of open source contributors (p. 6). Other more recent studies seem to bear out the findings von Krogh & von Hippel summarize.

Maas' case study (2009) focuses on determining the factors that motivate members of OSS communities to collaborate. He evaluated the Apache Cocoon open source development community, an established group of around 60 active developers who do a variety of work, including development, bug patches, and documentation. He used an online survey with 42 questions designed to reveal both demographical and motivational information about the community's members (p. 66).

The average age was found to be 31.2 years. Mass divided the workers into two distinct clusters – those 37 years or older with many years of technology experience, and those from 19-31 years old, with less experience. He found the first group less anxious to learn new technologies, but more actively involved in the project's day-to-day work. The younger members he found more interested in watching what is being done, and learning how to do it. This group was comparatively more interested in new technologies and "self-realization" (p. 68).

The questionnaire revealed that on average, each developer works 3.1 hours a week on the project, and a surprisingly high percentage (74%) are paid by their regular jobs to participate. A common characteristic of both groups was resistance to bureaucracy, which members find a "distraction" from software development (p. 67).

Maas concludes that a mature OSS project is able to attract highly trained experts in the field. Though members of Apache Cocoon project tend to work alone or with only a couple other developers, they have a great deal of trust in the skills of their peers, whom they expect to be highly skilled and altruistic (p. 69).

In their survey analysis, Lakhani & Wolf (2003) find that developers simply enjoy working on OSS projects, and while the experience may help further their careers, it is

primarily the creative and intellectual stimulation which drives them to participate (p. 3). The authors provide a fine literature review on intrinsic and extrinsic motivations (p 4-8). One interesting piece of data from the survey reveals that 87% of the respondents did not get paid directly for their work, and yet 55% reporting working on the project during their regular jobs. Of these, more than half indicated supervisor awareness, and "tacit or explicit consent" (p. 9).

Group Forums and their Role in Defect Tracking and Resolution

According to Ahmed et al. (2009), OSS does not follow proprietary vendors' "traditional life cycle of software development," and in general delivers software more quickly and depends on user involvement in "bug/defect detection." (p. 175). Discounting concerns about the quality of OSS, the authors use statistical analysis to conclude that the OSS community's use of group forums is more effective when it comes to tracking and resolving software defects than are propriety vendors' use of maintenance teams. This empirical study of 619 group forums and project defect tracking systems found a positive correlation between group forum messages and defect resolution. The authors conclude that in an OSS project, "defects are not simply accepted" but rather the project's "support network" collaboratively works to quickly resolve them. (p. 177). Closed software has a more traditional method of managing its software, and its "profit oriented vendors" generally assign maintenance teams to support maintenance releases (p. 174), while OSS has a "well-defined community with common interests" who in general contribute without "being employed, paid, or recruited." The authors assert that by having many interested developers working on a common project, the quality of the software is greatly improved and defects are resolved faster. (p. 174).

Evaluating and Implementing Open Source Solutions

Rafiq (2007) conducted an international study to measure the LIS community's perception of OSS adoption in libraries. He distributed his online questionnaire via international library discussion groups and listsrvs. He found that his respondents (68% from developing countries) were in general, positive about OSS in libraries. Concerns were mainly related to support, documentation, and implementation difficulty for library staff.

Writing for Educause, Trappler (2009) is also cautious about OSS, noting that this option can provide a "viable alternative," though it does not always save money or resources (para 1). He warns educators to be aware of the licensing agreements, and read them as carefully as they would a proprietary software license. Trappler warns that the complexities of some OSS may require "in-house support," and that it is not a "panacea" (para. 16). One advantage he notes is that it is much easier to try out OSS than proprietary software, and he encourages educators to always include OSS bids to expand competition with proprietary vendors.

According to Trappler, evaluating the "features, functions and maturity levels" of OSS products is vital to the selection process. If an OSS solution is chosen, an institution can benefit by becoming part of an OSS community, and sharing support functions. When "appropriately managed" OSS can be more effective and less costly than proprietary software (para 30-35).

Ven, Verelst & Manaert (2008) echo Trappler's cautions. Targeting ten Belgian organizations which had adopted OSS, their study recorded and analyzed face-to-face interviews with key employees. Points discussed were cost advantages, source code,

maturity, vendor lock-in, and external support. The authors conclude that organizations should not base their decisions to adopt OSS on what other organizations are doing or on the "claims in the literature." Rather, decisions should be made according to the specific needs and capabilities of the organization (p. 58-59).

Buchanan & Krasnoff (2005) examine whether open source can save school libraries time and money. They note that with the rising cost of proprietary software, and with the growing number of formats that don't translate across platforms, open source is being viewed as the more economical and flexible solution. (p. 1). But they also note that few educators have the expertise to maintain open source platforms, which tend to be more difficult to "install and configure, troubleshoot and upgrade" (p. 2), and the cost to hire outside consultants can negate other costs savings.

Digital libraries and Open Source

There are several studies that focus specifically on open source digital library software, and it is important to see if this subset of OSS has any specific evaluation concerns. Goh et al. (2006) conducted a comparative study of four OSS digital library packages and compiled a checklist to evaluate such programs. The framework they developed breaks down the products into five criteria for evaluation: (1) content management; (2) user interface; (3) User administration; (4) system administration; and (5) other requirements. Each of these major sections is further broken down into features, and each evaluator checks a box if the feature is available. Scores are weighted and tallied and then compared.

The recently published case study by Kucsma et al. (2010) is perhaps most relevant to this review. The authors worked on a project for METRO whose goal was to

build a directory of 30 digital collections. Three different digital collection management systems were considered: WordPress, Omeka, and CONTENTdm. Omeka was chosen because it was attractive, easy to install, extensible, and was compatible with Web and OAI-PMH standards. (p. 1). While METRO has a long history with CONTENTdm, it was not selected because, as the authors write, it lacks many of the "characteristics of a modern digital exhibition tool" (p.2). WordPress was not chosen because of its lack of collection building tools and because there was not enough time to write a plug-in.

The authors provide a good synopsis of Omeka's functionality and describe the plug-ins they employed in the process. In general they enjoyed their experience building this collection of collections with Omeka. The weaknesses they found were primarily related to Omeka's administrative functions. The authors found the "Item Add" function clumsy and time consuming, and they complained about the lack of support for a controlled vocabulary in the subject and tag fields. They admitted that these were fixable, but they hesitated to recommend using Omeka for a large scale project before the issues were addressed. Search and retrieval functions were found to be weak, by library standards. In general, they found Omeka's control over data to be "loose." (p. 8). They also found that documentation of the core processes had lagged behind. They felt the incomplete "Codex" was slowing down development of new plug-ins. Among its many strengths, the authors point out Omeka's exhibit building ease.

Literature Review Summary

Open source is a fascinating phenomenon. With roots in the 1960's, it firmly took a hold in 1983, when Richard Stallman started the GNU project whose objective, to create a totally open operating system, is now called Linux. Free Software Foundation

and Open Source Initiative, while differing philosophically, continue to collaborate and promote ways to freely share software, build project communities, and share development tasks. Open source projects evolve rapidly, though their success depends upon a self-sustaining community. Articles were reviewed which cover the movement's background, the demographics and motivations of members of open source communities, evaluation frameworks, digital library implementations, and the ways defects are tracked and resolved. Open source challenges include questions concerning support, documentation, and the ability of an institution's staff to take on administration roles. Advantages include superior software, complete control, extensibility, and community building.

Chapter III: Methodology

Overview and restatement of problem:

This chapter describes the methods used in this case study to collect and analyze data. The units of analysis outlined below form a framework to evaluate whether Omeka, as an open source web publishing alternative, can make the *Waterways* collection come alive. Developed and maintained in a manner common to most open source products, Omeka is at once representative and revelatory. It resonates many of the characteristics of open source. The issues uncovered in the process of migrating the *Waterways* collection to Omeka reveal both the advantages and the challenges of using open source.

Units of Analysis

The units of analysis of this study are: (1) Setup and Customization, (2) Importing Content, (3) Content Presentation, and (4) Administration. As Wildemuth (2009) notes, many methods of data collection can be used in a case study, and of those which she describes, direct observation, examination of physical objects, and analyses of existing documentation are employed (p. 54-55). Each of these methods is qualitative, but some triangulation may be achieved when applying alternative quantitative data collection methods (i.e. counting access points, recording the time it takes to import the postcards, and comparing the number of clicks required to conduct five, pre-defined searches). This study does not attempt a true comparative analysis of the CONTENTdm and Omeka collections, but instead addresses some of CONTENTdm shortcomings, as identified by the Waterways' administrators to explore how Omeka handles similar tasks.

Using a grounded theoretical approach, this study uses a recording instrument to collect and evaluate its findings. This study is an iterative process. It builds and

configures a website, and imports images into it. As problems crop up, processes are redone in different ways, according to the lessons learned. The Initial Evaluation Criteria describes the evaluation framework, which was purposefully been left open-ended for unanticipated findings, which required description and further investigation:

Initial Evaluation Criteria

1. Setup & Customization
    1.1. Initial Setup
        1.1.1. Download & Install the software
        1.1.2. Database Configuration
    1.2. Theme Considerations
        1.2.1. Theme evaluation & selection
        1.2.2. Designing the Waterways theme
    1.3. Plug-ins
        1.3.1. Selecting appropriate plug-ins
        1.3.2. Downloading and installing
        1.3.3. Configuration
2. Importing Content
    2.1. Reformatting the data exported from CONTENTdm
    2.2. Measuring the time it takes to importing items
    2.3. Problems?
3. Content Presentation
    3.1. Access Point Comparison
        3.1.1. Number of Access Points on CONTENTdm
        3.1.2. Number of Access Points on Omeka
    3.2. Retrieval Analysis (CONTENTdm vs. Omeka)
        3.2.1. Find postcard ID “H0070”
        3.2.2. Find postcards about “Canal boats”
        3.2.3. Find postcards from Oswego, NY
        3.2.4. Find postcards with Hudson River in title
        3.2.5. Find postcards with IDs like “H00*”
    3.3. Exhibit Building Functionality
    3.4. Slideshow Functionality
    3.5. Adding additional pages
4. Administration
    4.1. Editing Items
    4.2. Loading new items

The following four sections describes in more detail each of these units of analysis.

Unit 1 - Setup and Customization

The initial setup of an Omeka digital collection may be divided into three categories: setting up and configuring the software, selecting or designing a theme for the collection, and selecting and configuring the necessary plug-ins. Each of these tasks may require expertise and perseverance in an open source environment, where major criticisms include lack of documentation, difficulty to setup, and lack of support (Buchanan & Krasnoff, 2005). Alternatively, open source's rich culture of group forums and community support is often touted as one of its strengths (Ahmed et al., 2009; O'Reilly, 1999). What does the experience of setting up and configuring Omeka reveal about the product itself, as well as about open source?

The author's own experiences setting up Omeka is described in depth, and a content analysis of Omeka documentation performed. Questions in this unit include: Was the documentation sufficient for initial setup? What stumbling blocks (if any) were encountered? Were Omeka themes good enough to be used "out of the box" for the collection? How easy or difficult is it to use Omeka plug-ins?

Unit 2- Importing Content

Wildemouth (2009) maintains that to understand what is involved in a task, it is often necessary "to actually do the task" (p. 69). Much may be learned by actually importing the *Waterways'* postcards into Omeka. The task of such a migration can be broken down as follows: (1) converting the XML metadata from CONTENTdm to a CSV

file; (2) downloading the images from the collection manager's backup drives; (3) reformatting; and (4) the actual import process.

The images in the *Waterways* collection are cataloged individually. A postcard is not treated as a single item, but rather as two images. The corresponding metadata is likewise segregated. For the Omeka project, it was decided at the outset to treat each postcard as a single item which has two images, a front and a back. It was thus necessary to condense the data for each postcard from two records to one. Additional reformatting was necessary, including adding a "tags" field, and two image identifier fields that are necessary for the import process. The following two research questions were addressed: How was the reformatting accomplished? How long did it take?

Import functionality is very important to any content management system, and interviews with professors who have managed the *Waterways* collection for past classes reveal that this was one of the "sore" points of CONTENTdm. Batch files on numerous occasions were reported to have been lost after upload and Web support team was unable to find them (Perry, 2010). How does the experience on Omeka compare? How long did it take to import a batch of postcards? What problems were encountered?

Unit 3 – Content Presentation

This unit objectively compares the final product of this pilot project with the *Waterways* CONTENTdm collection. First it looks at access points. How many ways to retrieve postcards are available in each collection?

Additionally, five standard search questions were created to compare and contrast the two platforms' advanced search functionality. How do CONTENTdm and Omeka

handle common search requests? How many clicks are needed to accomplish the searches?

Unit 4 – Administration

This unit investigates Omeka's admin panels and evaluates their ease of use. Though the focus of this study has been on migration, it is important to appraise how Omeka allows single items to be edited. Some of the literature (Kucsma et al., 2010) highlights shortcomings, so it is important to assess Omeka's edit functionality.

Conclusion

The units of analysis described above serve to both explore and evaluate Omeka's potential as a web publishing tool. The findings described in the next chapter resonate with the characteristics of open source, as outlined in the literature review. By nature, case studies are particularizing designs, and their "lack of generalizability" [sic] may be seen as both as a strength and a weakness (Wildemuth, 2009, p. 53). By focusing on a single event, such as the migration of postcards to a new platform, a case study can describe and evaluate, test out hypotheses, and garner rich experience. Combining quantitative methods to a case study allows the researcher to triangulate findings, and several search exercises yielded interesting results on many levels.

Chapter IV - Findings

Background and Timeline

Work on the pilot project began in January, 2010. Four sample postcards were used in this exploratory phase. In early March, 2010, the metadata and the images were received, and by the beginning of April, all 587 postcards were imported. What remained, writing the text for the "Welcome" and "About" pages, and creating the exhibit, was done in the subsequent weeks. Though not yet complete, this pilot project demonstrates what can be accomplished in a relatively short time span using Omeka.

The following sections describe the findings for each of the units of analysis, as outlined in the previous chapter. Unanticipated results expanded the original evaluation framework, and the changes are seen in Appendix 1 – Final Evaluation Criteria.

1.0 Setup & Customization

1.1 Initial Setup

Downloading the latest version of Omeka (Omeka 1.1) was simple. As Kucsma et al. (2010) observes, it is one of the many open source "LAMP" applications, designed to run on the Linux operating system, using an Apache Web server, with MySQL as its database, and PHP as its scripting language. (LAMP is an acronym formed using the first letter of these four open source products.) While the product does run locally on a Mac (i.e. "MAMP," to extend the acronym), it does not run locally on Windows ("WAMP"). According to the group forum, there are no current plans to support Omeka locally on Windows.

Running software locally, using either WAMP or MAMP, allows a developer to use his or her own computer to step through the code for easier debugging. It also allows

for quick installation and evaluation without the trouble and expense of uploading to a Web host. A Mac computer was not available for this study, and running Omeka locally was not attempted. (As a side note, the Omeka group forum notes that installation using Mac's new operating system, "Snow Leopard" is tricky.)

Initial attempts to install Omeka 1.1 on a DailyRazor Web host were deceptively successful. Everything appeared to be functioning correctly until the *CSVImport* plug-in was installed. Then it was clear something was not working. After five days of back and forth correspondences on Omeka's group forum, the problem was diagnosed: the Web host needs to allow background PHP processes, and DailyRazor does not. (It seems many commercial Web hosts consider background tasks problematic, and block them.) After setting up an account with one of the Web hosts recommended by Omeka (*WebFaction*), the product was installed again.

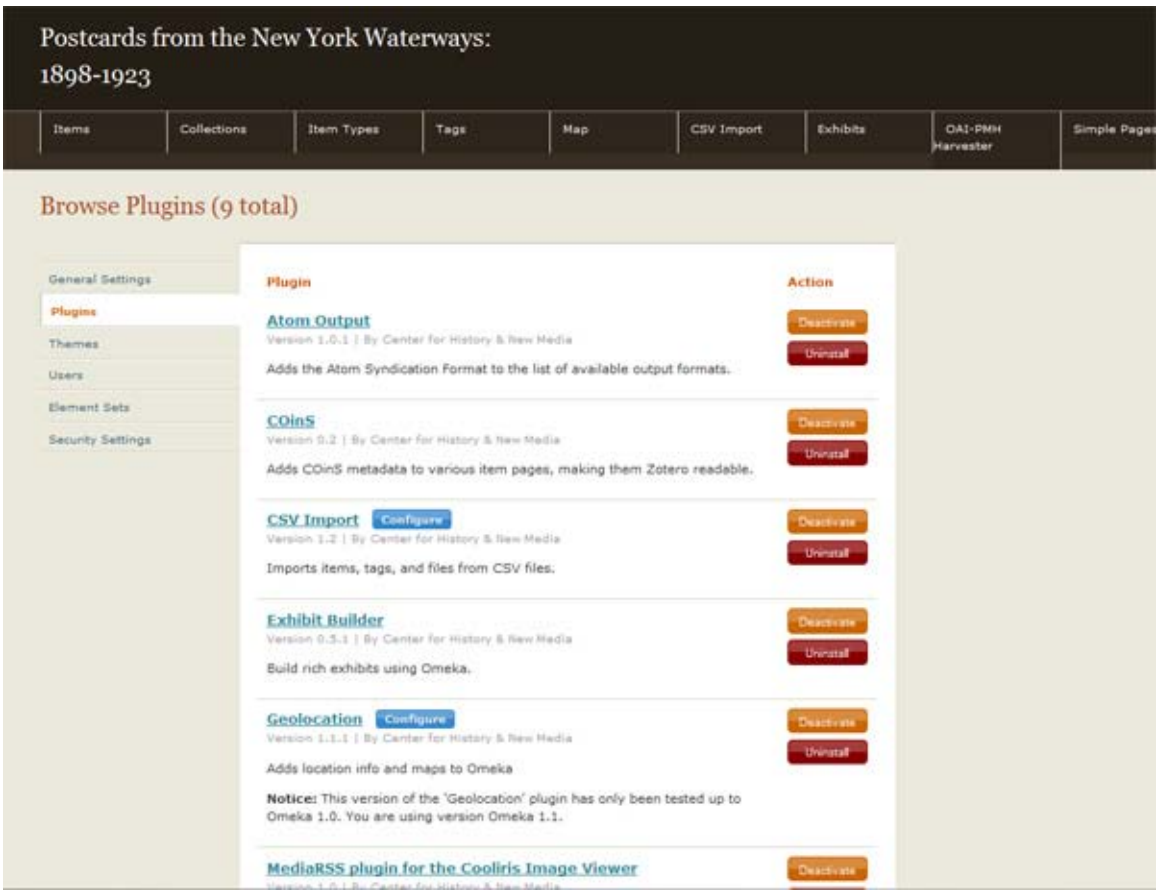
Installation on the new host was not initially successful, but the experience using the group forums proved invaluable. Three environment files (each called ".htaccess") in three separate folders need a one line change to work on this particular host. After searching the group forum, and reading the issues reported by others, the solution was found, and the software was fully functional, and a test import was successful.

Though frustrating, this experience served as a quick introduction to the open source model, where issues are reported on group forums, and help and support is provided by the dedicated community. This topic is further discussed in the "Support and Documentation" section, below. But first it is important to cover how Omeka is customized, using plug-ins and themes. Plug-ins extend Omeka's functionality, while themes change the look and feel of its user interface.

1.2 Plug-ins

Other than the problem previously described with the CVSImport plug-in and its Web host requirements, installing the requisite plug-ins was very straightforward. The process entails four steps: download the plug-in from the Omeka site, upload it to the Web host, install the plug-in using Omeka's admin panel, and configure it. See Figure 2 for a look at Omeka's plug-in administration.

Figure 1 - Plug-in Installation and Configuration Screen



The various plug-ins used in the Waterways collection are itemized below, along with annotations on each:

CSVImport. This plug-in allows for batch imports from files whose fields are separated by commas. This was the most problematic of the plug-ins (as previously described), as it requires a Web host which allows background PHP processes to execute. This plug-in was crucial to the project, and it will be more fully discussed in the "Importing Content" section.

Geolocation. By using *Google Maps*, this plug-in is able to geographically locate the items in a collection. A longitude and latitude is assigned to each item in the admin panel. This was a little tricky to configure, but adds a great deal to a collection. It is a good example of a "mash-up," where two or more Web 2.0 tools or platforms combine forces.

ExhibitBuilder. This plug-in allows users to create exhibits which place the items of a collection in context. It addresses Bullen's (2008) observation about most content management systems: they are good at storing and accessing images, but not very good at putting those images in context (p. 31). By using *ExhibitBuilder*, images may be selected from the collection, and pages constructed which in a sense "explain" an item, an historical period, or a trend.

SimplePages. This plug-in allows the user to construct ad-hoc pages, with HTML, PHP and JavaScript. In the Waterways collection, it was used to create all three "About" pages as well as the Slideshow. This was a very useful plug-in, and its name is perhaps misleading. *SimplePages* can do some very complicated things.

Coins. This is a very simple plug-in makes the page accessible to Zotero, another of George Mason University's projects. Zotero is a citation manager.

MediaRss. Another simple plug-in which enables RSS feeds. A user may click on an icon, get the feed in the browser, and copy the URL into his or her RSS Reader.

SocialBookmarking. This plug-in allows for pages to be shared using social networks like FaceBook, Delicious, and MySpace. Many options are available.

AtomOutput. This simple plug-in allows for feeds to be created in Atom format. To use it, the user just attaches "&output=atom" to the end of the URL and again copies that URL into his or her Atom Reader.

1.3 Theme Considerations

Unlike plug-ins, which extend Omeka's core functionality, themes control the way a collection looks on the screen. Omeka currently has only fourteen themes to choose from. They are meant to be interchangeable, and it is quite simple to change the appearance of a collection by changing its theme. The process of installing a theme is similar to that of a plug-in: the theme is downloaded from the Omeka site, uploaded to the Web host, and installed. The new release (Omeka 1.2) allows for configurable themes, which provide more freedom to easily change headers, footers and navigational methods.

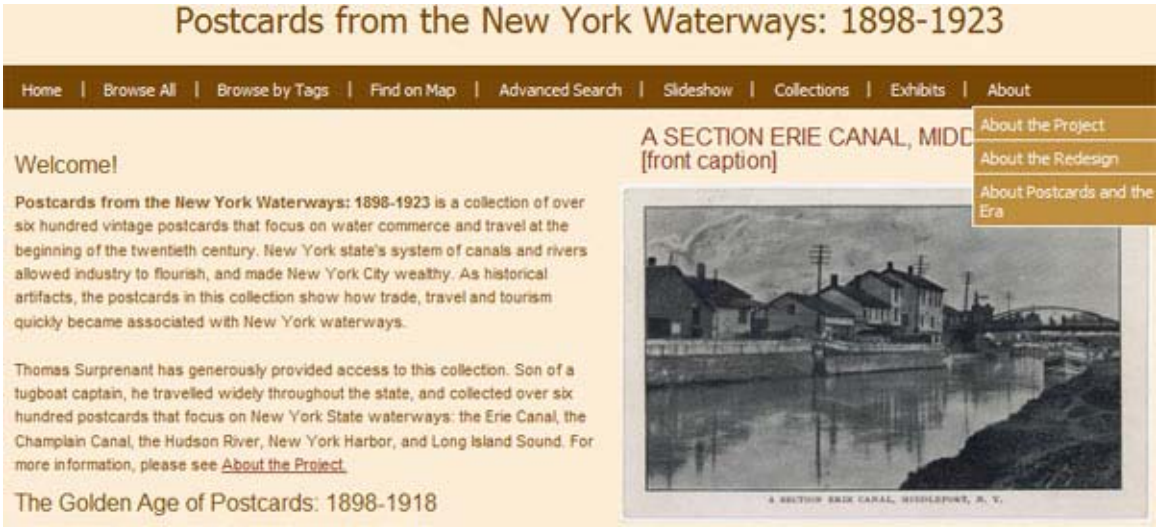
One of the shortcomings of CONTENTdm was its inability to treat a postcard as a single item, with a front and back image, and the goal of the new design was to enable this. This required developing a new theme, which also addresses another of CONTENTdm shortcomings -- its repetitive look. As Omeka becomes increasingly popular, this criticism may well pertain to it as well. Though Omeka (arguably) has a nicer look-and-feel better and better functionality, using one of the fourteen themes (many of which are quite similar) makes a site look like all the other Omeka sites.

The decision to treat a postcard as an object with two images had many unanticipated ramifications for this project, and they are discussed further in subsequent sections. Omeka can hold many images for each of its items, but its interface (or theme) only displays the first image. And that image is, for the most part, displayed as a square thumbnail. (Thumbnail size is configurable using "General Settings.") Postcards look very bad as square images. So the challenge was to create a unique-looking theme that displays each postcard as an information object with a front and a back image, and have those images not displayed in a square format. That challenge greatly complicated this project, but implementing such a theme proved invaluable in terms of understanding what extensibility means in open source products.

One of the most efficient methods of developing new themes, according to Omeka's website, is to take an existing theme and modify it. The "Emiglio" theme was chosen, and modifications began. In retrospect, it may have been wiser to select a less heavily formatted theme, such as "From Scratch" or "Minimalist," as some of the CSS was very much entwined, and rather complicated to modify.

Navigational is tightly bound in the themes. A single bar with drop-downs was wanted, but that was not available in the Omeka themes. This was the first experiment with extending Omeka's functionality: an open source, JavaScript navigation bar was found on the Internet and implemented in the header file of the theme. See Figure 2 for a screen shot of this:

Figure 2 - Navigation and Welcome Page



The Welcome page (also shown in Figure 2, above) was totally redesigned to provide space for explanatory text and one changing, clickable image, the front of a postcard, with its title and a portion of the description. This was accomplished by added a new PHP method called "display random postcard" into the theme's custom methods file. It was relatively easy to find a similar method in the Omeka core code, copy it and modify the parameters so that each time the screen was refreshed, a random front picture is displayed, not as a square thumbnail, but as a "plain" thumbnail.

The theme modified was heavily marked up with CSS (cascading style sheets), and much time was spent re-shuffling and fixing the borders, fonts and margins. A major lesson learned was to choose a theme with less formatting, and save time. Each page of the copied theme was divided down the middle, and that was not wanted in the new theme.

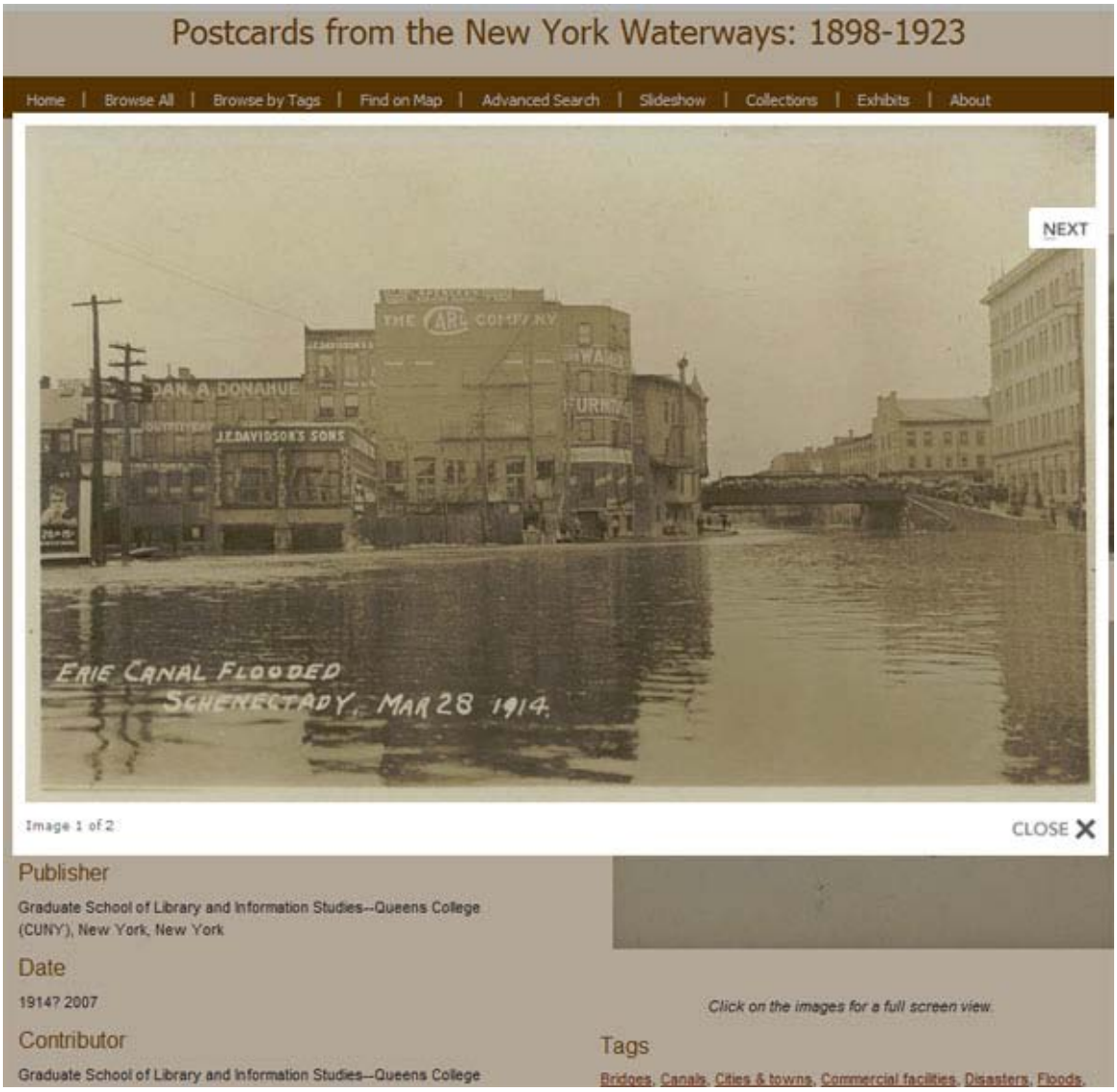
Item handling within the theme needed to be modified as previously discussed, so that two images were displayed side by side, front and back. Omeka is designed using a

Zend framework which subdivides the code in like-named files in different folders, and it took some time to understand the framework. The code is in three formats: PHP, CSS, and JavaScript. After much searching, the place in the PHP script was found which displays the item, and it was modified to show the second image. See Figure 3 for the results:

Figure 3 - Displaying a selected postcard showing both front and back sides

From the above screen, the user can click on each image to see a full screen image. Under Omeka 1.1, the image appears outside the Web site, by itself in the browser. When Kucsma et al. (2010) published their article and linked to their pilot Web site, it was observed that another plug-in called Lightbox2 was mentioned that does not appear on the Omeka site. Upon further research, it was discovered that this is actually a JavaScript package that can be configured to work with image display. After a little work, this package was implemented so that the image clicked opens up within the site's window (darkening the background), and a user can toggle between the two images, using the PREV and NEXT buttons, as shown in Figure 4:

Figure 4 - LightBox Image Handling



1.4 Support and Documentation

An often heard complaint about open source software is its incomplete documentation and lack of support (Rafiq, 2007; Trappler, 2009). Kucsma et al. (2010) add to that concern, noting that Omeka's Codex is a bit outdated and incomplete (p. 10). As an open source product matures, its community either builds or dissipates, and the product's success is held in the balance. Omeka appears to have reached a critical point in its lifecycle. It is gaining interest, and its group forums are quite active. There are a

generous number of screen casts that take users through the steps of installing and configuring plug-ins. Omeka's wiki allows the community to contribute to its "Codex" or core documentation.

Problems may surface when any software is installed, and it is important to measure how quickly and effectively issues are resolved. Both proprietary and open source face similar hurdles, though open source is often derided for its lack of documentation and support. Though proprietary software companies do not frequently use wikis to develop documentation, they have begun to understand the power of a user community, and have begun to adopt open source's group forum model that allows like-minded individuals to share their problems and solutions.

Conclusions about Setup and Customization

As with anything, if more time permitted, a more careful analysis of the existing code would have created better results. But it was quite easy to tweak the code, but this is probably not the approach open source community would want. Yes, a quick and dirty customization is possible, but to contribute the community, to build a theme that can be shared is more open source friendly.

2.0 Importing Content

Professor Perry provided an XML file of Dublin Core metadata for 630 postcards from the *Waterways* collections. Of these, 586 postcards were successfully imported into the new Omeka site. This section describes the process.

2.1 Reformatting the data exported from CONTENTdm

It was expected that reformatting would need to be done to combine two metadata records into one, but when the file was received, the extent of the re-formatting task was realized. For some reason, the CONTENTdm export combined description, contributor,

and technical information together into the Dublin Core description field, with no apparent separator. Work began to reformat the file.

First the XML file was converted into a comma separated file. Next it was loaded into a MySQL table, and PHP scripts were written to strip out the fields from description, which should be contributor and technical information. This was possible because much of the content of the two fields is constant (i.e. only the students' names change). Then the process of merging the front and back sides of the postcard metadata was done, using MySQL. Finally, a CSV file was generated, and uploaded to the particular folder where the CSVImport plug-in looks for uploads.

The major lesson learned during this phase was to make the size of the columns very large when creating the MySQL table. Numerous truncation problems were encountered due to the size of some of the fields. This again proved to be a very iterative process, as well as time-consuming, since the truncation is hard to immediately detect.

2.2 Image preparation

The 1,260 images were received on an external drive and down-loaded to a computer. Before uploading to the Web host, some initial checks were done to ensure the file sizes were reasonable (several .jpg files were actually .tiff files), and to ensure the names were correct. It was difficult to manually check such a large number of files, and in retrospect, perhaps a PHP script should have been written to identify missing images.

2.3 Importing the postcards

The import process involves the tedious task of mapping the fields on the CSV metadata file to the appropriate Dublin Core fields in Omeka, as well as identifying the image file names and tags. There is no way to save the mappings, so each time an import is submitted, the mapping must be done again. While the metadata mapping takes only a

little more than a minute, it is error prone, especially after doing it repeatedly. The CSVImport plug-in administration panel is used to for the mapping, and is shown in Figure 5.

In anticipation of problems, the CSV file was divided into six separate files, and the import proceeded in six batches. Unfortunately, Omeka has only two settings to handle when it encounters missing images on import. One option simply stops the process, and the other option simply continues the import. The second option proved problematic, since there was no way to figure out which postcards were only partially imported. A third option, to run the import in "check" mode, logging missing images, would have made the import process much smoother. So batches were submitted, expected to fail, then fixed and resubmitted until successful. Omeka has the option to "undo" a batch, which effectively removes the metadata records from the MySQL database, and removes the thumbnail, square thumbnail and full-size images it stores it respective folders. After a failed import, it was necessary to first "undo" the batch before starting over. Otherwise, duplicates appear in the collection.

Figure 5 - CSVImport Administration Panel

2.4 Measuring import time

Figure 6 shows the number of times each batch was submitted for import. Without errors, a batch of one hundred postcards took around twenty minutes to

complete. Total time for each batch to complete depended upon which postcard in the batch failed. (On one occasion, the ninety-ninth postcard failed in a batch, and the batch had to be redone.) The total time to complete the import was around ten hours, and was done in the course of three days.

Figure 6 - Time it took to import the postcards

Batch	Number of Postcards in Batch	Number of time Batch failed	Successful Imports	Failed Imports	Approximate Total Time (in Minutes)
1	100	6	94	6	105
2	100	5	93	7	100
3	100	5	92	8	90
4	100	4	91	9	110
5	100	3	93	7	85
6	130	3	124	6	105
Totals	630	26	587	43	595

The problems encountered were overwhelmingly due to missing images or incorrectly named images. These tended to occur in clusters, and as the process progressed, it was easier to anticipate and correct the problems prior to re-submitting the batch. One on occasion, special characters in the metadata stopped the process, and that needed to be corrected.

During the first batch, one of the imports just stopped without an error message. Apparently the background PHP process that does the import got hung up for some reason, perhaps due to a lost network connection. There was no way to “undo” the postcards which had been added, so it was decided (since this was just the start of the import process) to delete the collection, and start a new collection. This seemed to work, but later it was found that deleting the collection did not delete the postcards. At the end of the import, there were mysteriously fifty-seven more postcards than expected. There

appeared to be no other way than to go into the MySQL database and carefully delete the corresponding rows from the table. This was actually an interesting and relatively easy task, though it was done with some apprehension, as it was unclear what the repercussions would be. No attempt was made to delete the thumbnails, square thumbnails and full size images generated by this failed import, since the Omeka's naming convention made them very hard to identify.

Two other import issues bear mentioning. The first was related to a single postcard which appeared to have been imported correctly, but which displayed seven times on the Web site's detail screen. This one was a mystery, and perhaps related to some duplicate images that were unsuccessfully deleted. This postcard was manually deleted from the collection, using Omeka's Admin screen. The second issue had to do with special characters in the metadata. On several occasions descriptions mysteriously truncated upon display. Upon further investigation, it was discovered that a special character needed to be deleted from the metadata, again using Omeka's Admin screen, and the problem was solved.

The metadata for the forty-three postcards which failed import was stored on a file for later investigation. Following best practices, the *Waterways* collection has redundant back-ups, and these images will probably be found on another hard drive, and successful imports can add them to the collection.

Conclusions about Importing Content

Importing the metadata and the images of *Waterways* collection into Omeka was not expected to be an easy task, and the sections above testify to this time-consuming and sometimes frustrating process. Most of the hurdles encountered were not caused by

Omeka, but were due to the nature of a platform migration itself, where data needs be transferred to different formats, and images from external sources need to be carefully tracked and correctly named. Anywhere in the process, errors may occur.

But it would be incorrect to overlook some things which Omeka might do in the future to mitigate some of the difficulties. High on the list would be providing an import test-run functionality that would log missing images for an entire batch. Another would be a method to save import mappings, so that each time a batch is submitted, the mapping can be retrieved.

3.0 Content Presentation

3.1 Access Point Comparison

A quick count revealed that CONTENTdm has two access points with which to retrieve postcards: Browse All and Advanced Search. Omeka has four access points: Browse All, Browse by Tag, Find on a Map, and Advanced Search. A fifth access point, Slideshow, was added by finding and modifying an open source PHP slideshow program, and adding it to the Waterways Theme. (It could also be argued that Omeka's Exhibit Building functionality provides an access point to the collection, as does the randomly changing, clickable image on the "Welcome" page.)

3.2 Retrieval Comparison

The following five searches were conducted using both the CONTENTdm and Omeka versions of the Waterways collection, and keystrokes and content returned were recorded for analysis. The Omeka collection contains 43 less postcards than the CONTENTdm, and that needs to be held in account. But the metadata should be identical. The advanced search function was used in both platforms, though for question

three, Omeka users could have used the “Find on a Map” function. Both systems allow for field searching.

Figure 7 - CONTENTdm and Omeka Search Comparison

Search Question	CONTENTdm results	Omeka Results	ContentDB clicks	Omeka clicks
Find postcard – ID = H0070	1	1	4	2
Find postcards about “Canal boats”	137	114	3	2
Find postcards from Oswego, NY	28	12	5	5
Find postcards with Hudson River in title	176	79	5	5
Find postcards with IDs like "H007*"	21	N/A	4	N/A

3.2.1 Find postcard ID = H0070

Interestingly, keyword searching on both systems required exact word matches. Searches for H007, for example, yielded no results. CONTENTdm’s ID structure is different, so two searches were required, one for h0070ac1 (the front of the postcard) and h0070ac2 (the back of the postcard). While this is worth noting, the fundamental results of an ID search were identical on the two systems. Omeka’s keyword search was not case sensitive, while CONTENTdm was.

The first time advanced search is used on CONTENTdm, the user needs to deselect all the other METRO collections sharing the site, and then select the Waterways Collection. This accounted for the additional two keystrokes on the first search. Once a user does this, the system seems to remember which collection it is searching under.

3.2.2 Find postcards about “Canal boats”

Omeka required just two clicks for this search, by using Browse by Tags function. This brought back results from tag fields. A subject search brought back the same number of searches, but required an additional click. Although CONTENTdm returned more results, the result set was not as precise, and included some backs (even though the

search was within the “Subject front” field), which were double counted. In fact, some duplicates in the result set were observed.

3.2.3 Find postcards from Oswego, NY

Again, duplicates were noted in CONTENTdm. Both fronts and backs were retrieved, which was understandable, given the way the metadata is stored. The field search on Omeka, using “Coverage” was quite precise. It is worth noting that the “Find on a Map” access point could supplement (or replace) such a search.

3.2.4 Find postcards with Hudson River in title

The options available on field searches in Omeka are “contains”, “does not contain”, “is empty” and “is not empty”. In CONTENTdm, the terminology is “All the Words”, “The Exact Phrase”, “Any of the Words” and “None of the words.” So for Omeka, “contains” was selected, and for CONTENTdm, “any of the words” was selected. The search results were very close, considering the previously mentioned duplication in CONTENTdm.

3.2.5 Find postcards with ids like “H00*”

Omeka does not allow wildcard searches, and this correlate with Kucsma et al. (2010), which found Omeka’s search functionality not “library level” (p. 8). CONTENTdm was quite good with wildcard searches.

Conclusions about the Advanced Search Comparison

The advanced search functionality in both CONTENTdm and Omeka could both be better. Wildcard searches are not available in Omeka. The discrepancies in search results appear to be due to the different ways the metadata is stored. Omeka required slightly fewer clicks to get results, but CONTENTdm was slightly stronger, due mainly to its wildcard search functionality.

3.3 Exhibit Building

One of the reasons why Omeka appeals to museum, libraries, and archives is its ability to put a collection in context by building interpretative exhibits. The *ExhibitBuilder* plug-in provides eight different page layouts, which govern the way images and text appear on the page. Images may be searched for, and when found, dragged and dropped onto the layout. Exhibits may be created with units and sub-pages. It is fairly straightforward, though the interface requires a lot of scrolling and the way units, sections, and pages organized is clunky.

Again, due to the unique way postcards are handled in the collection (i.e. two images for each postcard, a front and a back), the back images of the postcards were not available to be dragged and dropped into the frames. This was disconcerting, especially for "The Golden Age of Postcards" exhibit, which discusses the various Acts of Congress that changed the format of the back sides of postcards. Fortunately, a work around was discovered. The plug-in allows standard HTML, and so the pages that display the backs of the postcards were done manually, and those which use the fronts where created using the drag-and-drop functionality. The drawback to this solution was that the back sides were not clickable like the front sides. PHP scripting does not appear to work within the exhibit frames, but certainly, given the time, this could be worked out.

The *ExhibitBuilder* plug-in requires its own separate theme, and a little experimentation with the available themes revealed that exhibits look out of place when their themes don't match the parent theme. To correct this, an exhibit theme was created which closely mirrors the main theme. This involved a lot of cut and paste, and within a couple hours work was complete.

"The Golden Age of Postcards" is a sample exhibit, and much work is needed to make it better, but it does demonstrate how Omeka can be used to put a collection in context, or to showcase a particular part of a larger collection.

3.4 Browse by Tag

The import process provides an additional, non-Dublin core field called "tags." This field was populated using what CONTENTdm holds in the subject field. Reformatted so that these words or phrases are separated by commas, the tags are imported into Omeka to become hyperlinks, and when clicked, perform powerful subject searches. This proved to be a very powerful access point for the collection, since the subject fields, now tags, were created using a the Thesaurus for Graphic Materials (TGM), and already passed validation when housed in CONTENTdm.

While tags work exceedingly well in this project, Kucsma et al. (2010) point out that Omeka does not support controlled vocabularies. The authors find this a big issue, since many variant terms may be used to describe an item, and the absence of such controlled vocabularies leads to what they call a "loose" control over data (p. 8).

3.5 Slideshow

Slideshow functionality is conspicuously missing from Omeka, and its absence provided an opportunity to evaluate the difficulty of extending Omeka's functionality. Normally in the open source model, a developer would create a plug-in and share it with the community, which would then test it, find any defects, and suggest enhancements (Ahmed et al., 2009). But due to lack of time (and expertise), this project did not attempt to create a plug-in, but rather found an open source slideshow called PHPSlideShow, written by Greg Lawler, and integrated it into a "simple" page using the *SimplePages* plug-in. The experience shows how extensible open source products like Omeka can be,

and what can be accomplished with a little work, and how mashing up open source products can create new functionality – in this instance, a new access point.

The slideshow function is by no means perfect, and not much time was spent polishing it, since time was short and expectations high that an Omeka slideshow plug-in would be developed soon.

3.6 Additional Context Pages

The plug-in *SimplePages* allows the user to construct many pages which provide additional information about the collection, its sponsors, and the project. This kind of information is not available in the CONTENTdm collection. Omeka allows the administrator to create many additional pages, which can be plain text, HTML, or even PHP scripts that interact with the collection itself and generate specific or random images, along with metadata such as titles and descriptions. As Perry (2010) notes, such context pages offer a good opportunity to "brand" a collection (para. 1). It also offers ways to give credit to a collection's owner and project team.

3.7 Find on Map

Geolocation is a very useful plug-in which adds a geographical access points to the collection. Though one of the trickier plug-ins to configure, it adds another dimension to *Waterways*, but its full implementation would require much work. CONTENTdm obviously does not have the latitude and longitude of the postcards, but even if it did, there is currently no way to import this information into Omeka. The only solution is to use the Omeka Admin panel, bring up each postcard, click on "Geolocation," find the postcard on the map and click to mark it. To pin-point the geographical location of all the postcards in the collection would be a large task best

shared by a group of people. For this pilot project, only around thirty postcards were geographically located. See Figure 8:

Figure 8 - Find on a Map

Conclusions about Content Presentation

Omeka's strength in creating attractive access points for a collection has no doubt contributed to the interest it is currently garnering. Though there is some missing functionality (i.e. a slideshow), the platform's open source model makes it very

extensible. Functionality may be added or modified to suit the needs of a collection. A comparison of the advanced search functionality yielded mixed results, and while neither platform's performance was outstanding, Omeka consistently required less time to generate search results.

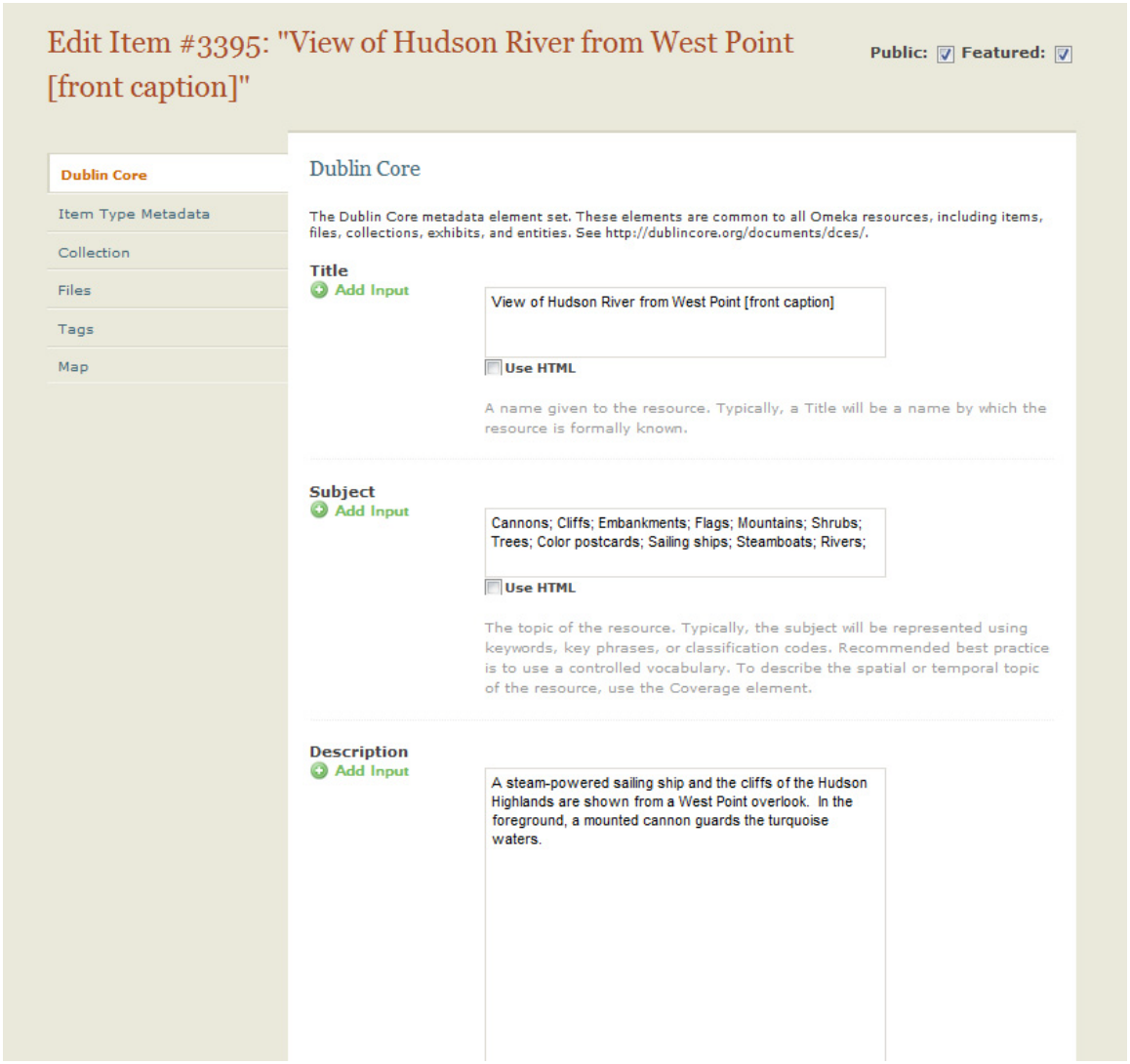
4.0 Administration

One obvious advantage over CONTENTdm is the total control over the collection that Omeka offers, and yet, as has been seen above, some expertise is required to put all the pieces together. This section investigates how the Admin panels serve to control the collection.

4.1 Adding and Editing Items

Omeka's import process described previously uses the *CSVImport* to ingest items from a comma separated file. Omeka also allows new items to be manually entered using the Admin Panel. And as noted previously, Omeka does not use a controlled vocabulary for any of these fields, although within the descriptions of each Dublin Core field, suggested thesauri appear. (A controlled vocabulary plug-in is currently being developed, as noted in Google's "Omeka Dev" group.) HTML is allowed by clicking on a checkbox. Another button allows multiple entries for each field to correspond to Dublin Core's schema. As was noted by Kucsma et al. (2010), adding and editing items on the Omeka's Admin Panel requires much scrolling. The page is lengthy, due largely to number of metadata fields and their potential size. Figure 9 shows just the first three fields of an item. The right navigation bar shows additional options to edit.

Figure 9 - Omeka Admin Panel for Editing Items



While this project focused solely on the item type Omeka calls the "still image," many other item types are supported, including document, oral history, moving image, sound, website, event, email, lesson plan, hyperlink, person, and interactive resource. Collections can hold many different item types, although the *CSVImport* plug-in does not allow a mixture of types in any one import.

4.2 Rights and Permissions

Delegating tasks is important and Omeka has limited support for privilege handling. On the Admin Panel, users may be added using the "Add a User" function. There are four roles available: Super, Admin, Contributor and Researcher. The Super can do everything. Both Super and Admin roles can change users' passwords if they have been forgotten. IDs and Passwords are mailed to users, and they may change them if desired. The Admin role can perform all the functions a Super can, except delete collections, and install and configure themes and plug-ins. Contributors can add items, edit and delete their own items, and tag any item. Researchers can see items that have not been made public yet. (Omeka Codex).

Conclusions about Administration

Though they require much scrolling, Omeka's Admin panels adequately control collections. Support for many types of items is provided. Add and edit functions will be strengthened when plug-ins are developed to hook into the various controlled vocabularies. Work is currently underway. Four user roles are available to help delegate permissions.

Summary of Findings

Migrating the *Waterways* collection to Omeka was an enlightening experience on many levels. The units of analysis revealed valuable information about Omeka which were found to often resonate with the topics discussed in the literature review. A "hands on" approach to research showed how group forums may be used to solve installation issues, how plug-ins and themes can be used in enhance functionality, and how the import process, by its very nature, is an iterative and sometimes frustrating ordeal. Comparisons between CONTENTdm and Omeka's access points and advanced search

functionality added some quantitative data with which evaluate content presentation. A look at the Admin functions provided new insight into aspects of Omeka that were not directly used in the migration.

Chapter V – Conclusions

The Open Source Model and Omeka

The open source model is an attractive alternative to proprietary Web publishing, but as the literature points out, adopting such a solution is not free (Trappler, 2009; Ven et al., 2008). Open source requires a level of expertise which may involve additional staff training, or under some situations, employing consultants to do the ground work of setup and customization. Additional expenditures of staff time may also outweigh any cost savings, but as some note (O'Reilly, 1999; Ahmed et al., 2009), the superior quality of the software more than compensates for the hours spent. Open source software by is by nature extensible and customizable, and if such an approach is used for web publishing, an organization assumes full control and responsibility over its collections. Success depends upon the willingness of staff members to take on new roles and challenges, and to build upon their expertise.

The process of migrating the *Waterways* collection to Omeka was useful on several levels. It provided a glimpse of an open source community hard at work developing a Web publishing tool. The George Mason University's Omeka team, with the collaboration of technical staffs from museums, libraries, schools, and archives, is building a unique web publishing platform capable of both storing and accessing digital collections, and putting those collections into perspective with interpretative exhibits.

The "hands on" approach used in this study also provided valuable insight into how open source projects may be judged. The evaluation framework expanded and contracted as issues cropped up in each of the four units of analysis. Setting up and customizing Omeka showed how group forums may be used to resolve issues. Problems

that were encountered in this unit of analysis echo the concerns and hesitancy to adopt open source noted in the literature (Goh et al., 2006; Ven et al.).

The import process revealed how certain features of an open source platform may be irritatingly underdeveloped, but with the input of the community, defects or shortcomings may be reported, addressed and ultimately resolved. CONTENTdm and Omeka access points and advanced search functionality were compared. Rich with Web 2.0 tools, Omeka clearly has more ways to access collections, but its search functionality, while quicker to use, has not yet reached "library level" (Kucsma et al., 2010, p. 8). Omeka employs plug-ins to extend functionality, and themes to control a collection's look-and-feel, and as the platform matures, it will be interesting to watch these develop. They are currently limited. There was unfortunately not enough time to spend on interactive exhibits, though two exhibits were started: while Omeka's effort to enable interpretative exhibits is admirable, the functionality still needs some polish.

The final unit of analysis dealt with Omeka's Admin functionality. Concurring with Kucsma et al. (2010), the study found that the amount of time needed to add items to collections was considerable, and the lack of a controlled vocabulary troublesome. (The "Omeka Dev" Google Group shows that at least one plug-in is currently in the works to integrate controlled vocabularies.) Omeka is able to delegate rights, so that an administrator can control what workers are permitted to do with a collection.

Having access to the *Waterways* collection provided a unique opportunity to experiment with Omeka to see how it competes with a proprietary platform such as CONTENTdm. Though still not yet mature, the platform is an exciting alternative that may be customized and extended to suit the needs of a collection.

Lesson Learned

Had everything proceeded smoothly in this project, this would have been a rather dull account of a very mature open source product. The fact that many issues were encountered, and that they resonated so much with the literature proved to be invaluable learning experience. This section highlights the major lessons learned.

Migrating digital collections across platforms is a complicated task, and organization, planning, and perseverance are vital. Iterative processes should be broken down into repeatable steps, which may be slightly modified to address the issues which crop up. Mash-ups with other open source products (LightBox, PHPSlideshow, Navigation Bar, etc.) are an effectively way to extend functionality. Group forums are incredibly useful when trying to figure out issues in open source.

Presenting an information object with two images does not fit many platform molds, and for Omeka, this implementation required a lot of work. Omeka works quite well "out of the box," and small modifications to themes are not that complicated. But major theme changes have ramifications throughout the code base, and one needs to be prepared to work around their effects. This was in particular noticed when using the *ExhibitBuilder* plug-in, which creates interactive exhibits. Only one image per item is available for selection, and either a plug-in modification or a workaround was required. Luckily, a workaround was found.

The overarching lesson learned pertains to open source itself. To fully embrace an open source project, one needs to be involved with and contribute to that project's community. This may be done by contributing to group forums, or by testing the software, reporting defects and tracking their resolution. Developers can write new plug-

in and themes. Other ways to contribute include adding documentation to the project's wiki, suggesting future enhancements, blogging about the project and its upcoming releases, and showcasing other collections. The success of an open source project depends upon the dedication of its community (O'Reilly, 1999), and the necessity of contributing in this way proved to be the major lesson learned in this study.

Additional Research Opportunities

This paper and the accompanying Website were created in one semester, and time did not allow for such important tasks such as a usability study and a user satisfaction survey. Without these, it is difficult to measure the level of success or failure of this project. Also worth noting is that Omeka is a moving target – towards the end of this study a new version was released, and researchers can fully expect the platform to continue to evolve, making evaluation more complicated.

Expanding on this content analysis, a usability study could examine log results to determine which access point is used most often. Additionally, a selected group of users could be assigned a controlled list of tasks, and the computer logs could reveal the amount of time required to find the object on both the old CONTENTdm site and the new Omeka site.

A survey could be constructed to gauge relative satisfaction with each site. Rather than having the control group compare the two sites, each participant could judge one site, and then the two sub-groups results could be compared.

And finally, a focus group containing interested parties could meet to discuss further steps, enhancements, and concerns with the new Omeka site.

Acknowledgements

Thanks go to Professor Claudia Perry for suggesting this research project and providing access to this rich collection of postcards. Her ideas were inspirational. Thanks also go to Professor Colleen Cool for all the research ideas and guidance she provided, and to Professor Thomas Surprenant for his interest in postcards, and to Walter Valero for first recognizing these postcards' digital potential.

Appendix 1 – Final Evaluation Criteria

1. Setup & Customization
    1.1. Initial Setup
        1.1.1. Web Host Considerations & Selection
        1.1.2. Download & Install the software
    1.2. Theme Considerations
        1.2.1. Theme evaluation & selection
        1.2.2. Designing the Waterways theme
    1.3. Plug-ins
        1.3.1. Selecting appropriate plug-ins
        1.3.2. Downloading and installing
        1.3.3. Configuration
    1.4. Support and Documentation
        1.4.1. Using the Codex
        1.4.2. Interacting with the Omeka Community
            1.4.2.1. Using Group Forums
            1.4.2.2. Using the Wiki
            1.4.2.3. Using the Screencasts
2. Importing Content
    2.1. Reformatting the data exported from CONTENTdm
    2.2. Image preparation
    2.3. Measuring the time it takes to importing items
    2.4. Problems?
3. Content Presentation
    3.1. Access Point Comparison
        3.1.1. Number of Access Points on CONTENTdm
        3.1.2. Number of Access Points on Omeka
    3.2. Retrieval Analysis (CONTENTdm vs. Omeka)
        3.2.1. Find postcard ID “H0070”
        3.2.2. Find postcards about “Canal boats”
        3.2.3. Find postcards from Oswego, NY
        3.2.4. Find postcards with Hudson River in title
        3.2.5. Find postcards with IDs like “H00”
    3.3. Exhibit Building
    3.4. Slideshow
    3.5. Find on Map
    3.6. About pages
4. Administration
    4.1. Editing Items
    4.2. Loading new items

References

Ahmed., F., Campbell, P. Jaffar, A. & Capretz, L. (2009). Defects in open source software: The role of online forums. *Proceedings of World Academy of Science: Engineering & Technology*, *40*, 174-178. Retrieved March 1, 2010, from Wilson Web.

Buchanan, K., & Krasnoff, B. (2005). Can open source software save school libraries time and money? *Knowledge Quest*, *33*(3), 32-34. Retrieved March 15, 2010, from Wilson Web.

Bullen, A. (2008). The 'Long Tale': Using Web 2.0 concepts to enhance digital collections. *Computers in Libraries*, *28*(9), 31-35.

Digital Collections. (n.d.). Retrieved March 6, 2010, from http://www.metro.org/index.php?option=com_content&view=article&id=52&Itemid=236

Goh D., Chua, A., Khoo, D., Khoo,E. , Mak, E. & Ng, M. (2006). A checklist for evaluating open source digital library software. *Online Information Review*, *30*(4), 360-379. Retrieved March 1, 2010, from EBSCO Host.

Free Software Foundation. (2009). Retrieved March 6, 2010, from http://www.fsf.org/about/staff/

History of the OSI | Open Source Initiative. (, n.d). . Retrieved March 7, 2010, from http://www.opensource.org/history

Kucsma, J., Reiss, K., & Sidman, A. (2010). Using Omeka to build digital collections: The METRO case study. *D-Lib Magazine*, *16*(3/4). Retrieve April 15, 2010 from http://www.dlib.org/dlib/march10/kucsma/03kucsma.html

Lakhani,K. & Wolf, R. (2003 9). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. Retrieved March 1, 2010, from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=443040

Maass, W. (2004). Inside an open source software community: Empirical analysis on individual and group level. *Proceedings of the 4th Workshop on Open Source Software Engineering.* Retrieved from http://74.125.155.132/scholar?q=cache:AqdNOgJyV0UJ:scholar.google.com/+open+Software+Web+OR+publishing&hl=en&as_sdt=20000000001&as_subj=soc

Omeka Codex. (2010). Retrieved March 7, 2010, from http://omeka.org/codex/

Omeka Project. (2010). Retrieved March 7, 2010, from http://omeka.org/about/

O'Reilly, T. (1999). Lessons from open source software development. *Communications of the ACM*, *42*(4), 32-37.

Postcards From New York Waterways: 1898-1923. (2010). Retrieved April 25, 2010 from http://www.pcnyw.net

Perry, C. (2010). About CONTENTdm and the Waterways Collection. Retrieved April 25, 2010 from http://www.pcnyw.net/CONTENTdm

Rafiq, Muhammad. (2009). LIS community's perceptions towards open source software adoption in libraries. *International Information & Library Review*, *41*(3), 137-145. Retrieved March 9, 2010, from Wilson Web.

Stallman, R. (1983). The Free Software Definition - GNU Project - Free Software Foundation (FSF). Retrieved March 7, 2010, from http://www.gnu.org/philosophy/free-sw.html

Stallman, R. (2002). Linux, GNU, and freedom - GNU Project - Free Software

Foundation (FSF). Retrieved March 6, 2010, from http://www.gnu.org/philosophy/linux-gnu-freedom.html.

The Open Source Definition | Open Source Initiative. (n.d.). . Retrieved March 20, 2010, from http://opensource.org/docs/osd

Trappler, Thomas J. (2009). Is There Such a Thing as Free Software? The Pros and Cons of Open-Source Software. *EDUCAUSE Quarterly*, *32*(2).

Valero, W. E., Perry, C. A., & Surprenant, T. T. (2007). History on a postcard. *Library Journal*, *132*, 6-9. Retrieved March 20, 2010, from Wilson Web.

Ven, K., Verelst, J., & Mannaert, H. (2008). Should you adopt open source software? *IEEE Software*, *25*(3), 54-59.

von Krogh, G. & von Hipper, E. (2003). Special issue on open source software development. *Research Policy*, *32*(7), 1149-1157.

Wildemuth, B. (2009). *Applications of social research methods to questions in information and library science*. Westport Conn.: Libraries Unlimited.

Williams, S. (2002). Free as in Freedom: Richard Stallman's Crusade for Free Software. Retrieved March 6, 2010, from http://oreilly.com/openbook/freedom/

access points, 6, 8, 9, 21, 25, 44, 50, 51, 55, 57
admin panel, 40, 50, 52, 54
advanced Search, 25,44, 47,51,55,57
Apache Cocoon, 14, 15
Apache Web server, 27
archives, 9, 47, 56
AtomOutput, 32
background PHP processes, 28, 30
Browse All, 44
closed software, 16
Codex, 19, 37, 54, 61
Coins, 31
collaboration, 6, 56
collection management, 8, 18
communal hacker culture, 11
community support, 23
configurable themes, 32
configuration, 5, 22, 29, 61
controlled vocabularies, 49, 54, 57
copy-left, 11
copyright, 7
cost savings, 56
CSS, 33, 34, 35
CSVImport, 5, 28, 30, 39, 40, 41, 52, 54
defects, 15, 19, 49, 57, 59
Digital Metro New York, 7
digital repositories, 7
documentation, 3,12, 14, 16, 19, 21, 23, 29, 37, 38, 59
Dublin Core, 38, 39, 40, 52
edit functionality, 25
ExhibitBuilder, 31, 47, 48, 59
Find on a Map, 5, 44, 45, 50
Free Software Foundation, 11, 19, 62, 64
funding, 7
gated open-source community, 12
Geolocation, 31, 50
George Mason University, 9, 31, 56
GNU Project, 11, 63, 64
Google Maps, 31
group forums, 15, 23, 28, 29, 38, 54, 57, 59
HTML, 31, 47, 49, 52
import process, 24, 40, 42, 48, 52, 55, 57
Importing Content, 3, 4, 21, 22, 23, 30, 38, 43, 61
information object, 33, 58
installation, 5, 28, 29
intellectual stimulation, 15
interactive resource, 53
interpretative exhibits, 6, 47, 56, 57
item types, 53
JavaScript, 31, 33, 35, 36
keyword searching, 45
lack of support, 19, 23, 37
LAMP, 27
lessons learned, 2, 10, 22, 58
libraries, 3, 7, 9, 16, 17, 47, 56, 62, 63
Lightbox, 36
Linux, 19, 27, 64
Mac, 27, 28
mapping, 40, 44
MediaRss, 31
MediaWiki, 8
metadata, 6, 7, 24, 27, 38, 39, 40, 42, 43, 44, 45, 46, 47, 49, 52
motivations, 14, 15, 19
moving image, 53
museums, 9, 56
MySQL, 27, 39, 40, 43
navigation bar, 33, 52
OAI-PMH, 18
OCLC, 7
Open Source Initiative, 11, 19, 63, 64
open source software movement, 11
oral history, 53
permissions, 4, 54
PHP, 27, 31, 34, 35, 39, 40, 42, 44, 47, 49
proprietary software, 10, 11, 16, 17, 38
reformatting, 24, 39
research questions, 24, 25
rights, 4, 54
scrolling, 47, 52, 54
shortcomings, 21, 25, 32, 57

SimplePages, 31, 49
slideshow, 4, 6, 23, 31, 44, 49, 51,61
Snow Leopard, 28
SocialBookmarking, 32
software hoarding, 11
source code, 11, 12, 13, 17
special characters, 42, 43
staff time, 56
still image, 53
subject search, 46
support, 3, 29, 37, 54, 61
TGM. *See* Thesaurus for Graphic Materials
theme, 22, 23, 32, 33, 34, 38, 48, 58, 61
Thesaurus for Graphic Materials, 48
thumbnail, 7, 33, 34, 40
timeline, 3, 27
usability study, 59, 60
user interface, 9, 18, 29
user satisfaction survey, 59
vendor lock-in, 17
Web 2.0, 6, 8, 31, 57, 62
Welcome Page, 5, 33
wikis, 38
WordPress, 8, 9, 18
XML, 24, 38, 39
Zend, 34
Zotero, 31