Digital content management: the search for a content management system

Yan Han

The author

Yan Han is Systems Librarian, University of Arizona Library, Tucson, Arizona, USA.

Keywords

Content management, Operating systems, Preservation, Information systems, Digital storage

Abstract

Digital content management system is a software system that provides preservation, organization and dissemination services for digital collections. By adapting the systems analysis process, the University of Arizona Library analyzed its needs and developed content management system requirements for finding a suitable information system that addresses the increasing needs of digital content management. Dozens of commercial and open source candidates were examined to match against the requirements. This article provides detailed analysis of three major players (Greenstone, Fedora, and DSpace) in key areas of digital content management: preservation, metadata, access, and system features based on the needs of the University of Arizona Library. This paper describes the process used to analyze and evaluate potential candidates and includes results of analysis to illuminate the process.

Electronic access

The Emerald Research Register for this journal is available at
www.emeraldinsight.com/researchregister

The current issue and full text archive of this journal is available at
www.emeraldinsight.com/0737-8831.htm

Library Hi Tech
Volume 22 · Number 4 · 2004 · pp. 355-365
 · ISSN 0737-8831
DOI 10.1108/07378830410570467

The need for digital content management

The University of Arizona Library has developed quite a few interesting digital projects, such as Little Cowpuncher and Southwest projects. The *Little Cowpuncher*[1] is a mimeographed school newspaper written by Anglo and Mexican-American ranch children from 1932 to 1943 in Southern Arizona. The Southwest projects[2] consist of multiple unique southern Arizona collections such as South Arizona music and folk arts. These initiatives reflect unique southwest history and attract thousands of visitors around the world. At the same time we are the victims of our success, as each digital project was initiated and was completed on a case-by-case basis. These projects were created with different tools using a variety of technologies such as HTML, XML and PDF, and are not based on a cohesive approach. Furthermore, the library needs to store and publish external content to meet our customers' needs.

The current situation poses technical, financial and management problems. From the view of library technical staff, this approach requires various hardware and/or software to mount the content, which requires them to master a variety of computing technologies to do the routine management such as backup and updates, and thus increases job difficulty and workload. From the view of financial analysts, it is difficult to understand and track costs related to a project. This makes it difficult to predict the cost of future projects. From the view of management, distributed storage and individual access to the content also results in difficult issues in system management. For example, there are system security questions such as "Who should access what? Who should belong to the administrator group?", and network management questions such as "What bandwidth will this server provide? How much storage will that server need?" As few of the digital initiative activities were pursued in a coordinated manner, the absence of overall coordination and planning resulted in:

- the lack of consistent digital preservation standards to ensure the projects' accessibility in perpetuity;

The paper is based on the work of the content management subgroup at the Digital Library and Information Systems Team at the University of Arizona Library. The author would like to thank Krisellen Maloney for her support and advice for both the project and this paper as well as Paul Bracke's advice. The author would also like to thank other subgroup members, Jeremy Frumkin, Warner Onstine, Kevin Chen, and Eulalia Rocl, for their great work in drafting, organizing the CMS requirements, and evaluating candidates.

Library Hi Tech
Volume 22 · Number 4 · 2004 · 355-365

- the lack of uniform access protocols to ensure that the material can be readily discovered and shared by users and systems;
- the lack of consistent authorization and authentication methods to enforce access control;
- an inability to reuse available content; and
- an inability to monitor and predict costs associated with digital projects.

While there are escalating demands for digital access to information resources, the library continues to expend considerable resources providing access to the digital projects. We are increasingly aware that we need a way to manage the content effectively in terms of preservation, organization and dissemination. The implementation of a content management system (CMS) will provide the following advantages:

- *Improved information accuracy.* The quality of the information will be improved, as a centralized CMS provides accurate and up-to-date documents. A centralized CMS, which provides one location to store all the content, achieves some features of file control systems such as Unix's CVS. Content developers can use the CMS for record keeping and collaboration.
- *Increased flexibility.* The content can be repurposed in a variety of formats at desired time periods. In addition, the content can also be searched in a centralized place, allowing users and systems to easily find and access information.
- *Enhanced system management.* This approach will reduce network management and system security issues discussed above. Regarding the network management questions, system administrators will take care of network management for only one system. System security generally consists of authentication and authorization. Authentication means the process of a system validating a user's logon information, while authorization is defined as the right granted a user to use a system (Microsoft, 2002). A CMS also reduces system administrators' system security work, as it can be integrated with an external authentication service such as campus-wide centralized authentication LDAP (Lightweight Directory Access Protocol) for validating a user's identity. The integration will seamlessly combine the external authentication service with the system's built-in authorization. At the same time, the integration will also connect the system with the centralized authentication system. As a result, the integration will not only create an easy-to-use system so that a user only needs one logon for all systems, but also result in an easy-to-maintain system by eliminating separated authentication services so that a system administrator uses one database to manage all users' logon information.
- *Reduced maintenance and cost.* The implementation of a CMS will result in elimination of previous supporting systems on a case-by-case basis, which in turn allows us to manage the projects in a cost-effective manner.

A CMS is a fairly new concept. No standards were officially published for what a CMS should be and what the critical requirements a CMS should address. In information technology literature, most discussions of content management focus on managing comprehensive web sites and/or web collaboration tools. The exact definition of "content management" tends to be ambiguous due to the phrase having different meanings across various subject disciplines. Boiko defines content management as a process of collecting, managing and publishing content (Boiko, 2002, p. 67). A knowledge management company states that CMSs are a key way of managing and delivering business knowledge[3].

Based on the digital work at the University of Arizona Library, our needs are different from this definition. Our CMS should provide tools and support for preservation, control and dissemination of locally developed documents and external content. It must be cost-effective at the same time. We began a systems analysis and design process to select a CMS that would provide a foundation for our digital content management process.

System analysis of a content management system

Defining a content management system

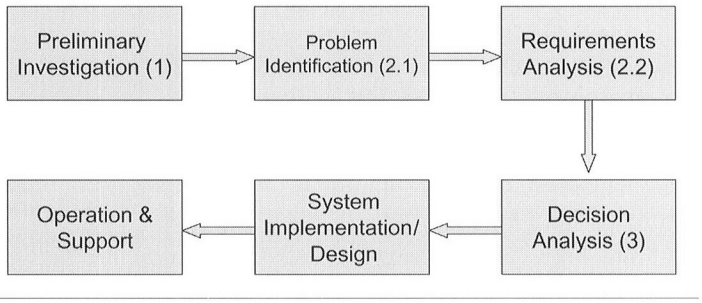
We adapted the systems analysis approach (see Figure 1), which generally consists of preliminary investigation, problem identification, requirements analysis, decision analysis, system implementation or design and finally operation and support (Whitten, 2001, p. 85).

A group was formed to include people with expertise in computer technology, metadata, and systems analysis. The first step was to analyze the problem and define the overall goals for a content management system. Based on our library setting, our ideal CMS will provide:

- a stable and extensible foundation to provide preservation and access services for end users;
- a stable and extensible foundation to provide preservation, access and information sharing services for other information systems;

Library Hi Tech
Volume 22 · Number 4 · 2004 · 355-365

Figure 1 Systems analysis flow chart (numbers in brackets are the sections designated for the phases)



- a long-term strategy for preserving, organizing, and disseminating locally developed and external content;
- a long-term strategy for preserving, organizing, and disseminating metadata associated with the content; and
- a stable and extensible foundation to support re-useable content.

Our ideal CMS is an information management system that preserves, organizes, disseminates and manages locally developed documents and external documents with associated metadata. It will be one of the library's fundamental systems to provide services to fulfill users' and systems' needs. The system will manage growing digital content as a long-term solution and supports the shift from a physical library to a hybrid of physical and digital libraries in a cost-effective manner.

Developing content management requirements

The next step was to develop our CMS requirements that fulfill the needs of the University of Arizona Library. A requirement can be either a functional requirement or a non-functional requirement. A functional requirement addresses one of our needs for a system in content management areas such as preservation, while a non-functional requirement defines a constraint such as costs and team skill sets that impact the selection of a particular system. At the same time, we realized that each requirement should not be weighed the same. In other words, some requirements, which are critical to the success of the system, should weigh more, while some requirements, which will not affect the system's main functionality if missing, should weigh less. We distinguished them using words "MUST", "SHOULD" and "DESIRABLE" respectively.

To help us understand areas in which the requirements address, the functional requirements were grouped into four major categories based on their nature. These categories are:

(1) *Organization requirements*. This category is comprised of metadata, content, and other sections, which mainly address how to organize the content and its associated metadata. For example, the requirement "each digital object must have one or more associated metadata records" describes the relationship between content and metadata.

(2) *Presentation requirements*. Presentation requirements tackle a system's look-and-feel and statistics requirements.

(3) *Access requirements*. Access requirements include both internal and external accessibility, as well as providing security accessibility. The internal accessibility addresses users' needs to access content and metadata. For instance, the requirement "users should be able to browse digital objects" means that the system should provide browsing service to users. The external accessibility tackles other systems' needs to access content and metadata. For instance, the requirement "the system must provide Z39.50 access" means that the system must allow a Z39.50 system to retrieve data from the system. The security accessibility defines authentication or authorization services. For example, the requirement "the system must use authentication to verify users with campus security system" addresses authentication issues.

(4) *Preservation requirements*. Preservation requirements consider storage, backup and long-term preservation issues for content and metadata.

Non-functional requirements were also drafted which do not address any functionality that the system should perform, but are factors that affect our decision making. Table I illustrates "Economy factors" such as system hardware/software cost and staffing cost. Appendix 1 includes a full version of these functional and non-functional requirements.

The search for a content management system

Preservation, metadata and access are key areas for the success of digital content management. The cost of developing our ideal system was too expensive to be considered. As there were quite a few candidates in the current industry, the requirements enabled us to focus on the search for a CMS through commercial and open source products. A total of 17 content management systems were identified. To narrow the scope, we preformed a preliminary evaluation of all content management systems. At this level we asked broad questions regarding digital content management, including questions related to preservation such as "Does the candidate define any digital preservation strategy?," questions relevant to metadata such as "Does the candidate support

Library Hi Tech
Volume 22 · Number 4 · 2004 · 355-365

Table I Non-functional (economy) requirements

3. Economy	
N3.1	The system (hardware and software) must be cost effective compared to similar systems
N3.2	The system maintenance (staff, hardware and software) must be included in cost assessment
N3.3	We must measure the cost of the skill sets within the library against that of obtaining the lacked skill sets
N3.4	The system must be provided continuous support from the vendor or groups

standard metadata such as Dublin Core?," and questions relevant to access such as "Does the candidate provide OAI or Z39.50 support?" In our web collaboration area, each candidate system was listed with its name and URL, the URL of its test version if available, a brief description, and an internal comment area dedicated to its ability to be a potential content management system. Each member was asked to provide his/her opinions on each candidate as well as to vote whether this candidate should go into the second round of selection. Each member could vote using " + 1" (strongly agree that we should examine this candidate), "0" (no opinion and hand over the decision to other members), and " − 1" (strongly disagree that we should examine this candidate). This process allowed us to speed up the first round selection. Within a few weeks we narrowed down the dozen candidates to three (Fedora, Greenstone, and DSpace) for a closer examination.

3 A closer look at three candidates

The second round selection allowed us to look at each candidate indepth and analyze its functionality and uniqueness in all areas including preservation, metadata and access. Based on our environment, each candidate was evaluated in its capability to address the issues outlined in our requirement sheet (Appendix 1). Each requirement in the requirement sheet was marked with either "\/" or "\X" or "X". A "\/" symbol means that the candidate has the ability to address this requirement, and a "\X" symbol indicates that the candidate has partial ability to handle this requirement or the candidate must be customized to fulfill this requirement; while a "X" states that the candidate is unable to meet the requirement. Please refer to Appendix A for a full version of the capability of each candidate.

Greenstone and DSpace received similar scores, meeting our functional requirements (see Appendix 1), and Fedora lacks some critical functions that are important to us. The feasibility of each candidate was further analyzed based on the above result and non-functional requirements. Our feasibility analysis (see Appendix 2) consisted of four major categories: Operational; Technical; Schedule; and Economic. Operational analysis describes how well each candidate system work for our environment, three candidates' scores were calculated from the requirements analysis of Appendix 1. Technical analysis shows the maturity of each candidate's technical solution, and our team's skill sets to customize and maintain the candidate. For instance, Greenstone is mainly programmed in C++and Perl, while Fedora and DSpace are designed with Java. Our team members have Java programming experience, but few having knowledge of C++and Perl. Therefore, Fedora and DSpace scored higher than Greenstone in development/maintenance cost. Schedule analysis is the time needed to implement the candidate, and economic analysis demonstrates soft/hardware requirements and development/maintenance costs. The detailed assignment of the feasibility analysis can be found in Appendix 2.

An in-depth look at the three systems resulted in some interesting findings, including the approaches these systems take towards key features of digital content management. We analyzed the advantages and the disadvantages of the three candidates in respect to five critical areas in digital content management:

(1) *Preservation issues*. We analyzed a candidate's preservation approach in a hierarchical order:
   - Does the system preserve the file's original identities such as its name, size and created date?
   - Does the system have any data integrity check for a file?
   - Does the system have any migration strategy as a long-term solution?

(2) *Metadata issues*. Generally speaking, a CMS supports at least one metadata set. Therefore, we examined a system in "Does the system support multiple metadata sets?" This feature is critical in digital content management, since a particular collection may be only interested in a specialized metadata schema. Museum collections, for instance, tend to use SPECTRUM (Standard ProcEdures for CollecTions Recording Used in Museums) metadata to describe museum focused information such as acquisition, loans history, and rights.

(3) *Access issues*. We analyzed a system's internal and external accessibility as well as its system security, including:
   - Does the system currently support persistent URLs to allow users and systems to access an object?

Library Hi Tech
Volume 22 · Number 4 · 2004 · 355-365

- Does the system's persistent URL method handle an object's change in location and state in the future?
- Does the system provide OAI-PMH and Z39.50 support to allow other systems to access the content?
- Does the system provide enough authorization methods to support system security management?

(4) *System features*. We looked at software evaluation criteria (Mandelbaum, 1992; Walton and Taylor, 1986) such as hardware and software, software quality, training and documentation, including:

- How good is the system's installation process?
- Does the system have enough document and training materials?
- How good is the system's search capability? Is the system's search engine effective and efficient?
- Are there any known issues/bugs in the system?
- What are the system's unique features?

(5) *Other noticeable issues*. We considered other issues related to our team skill sets, including whether our group has enough skills to customize the system?

Greenstone

Greenstone[4], produced by the New Zealand Digital Library Project in the Computer Science Department at the University of Waikato, was developed and distributed in cooperation with UNESCO and the Human Info NGO. It is a suite of multilingual software for building and distributing digital library collections, including organizing information and publishing it on the internet or on a CD-ROM. Greenstone, original released in 2000, is an open source software under the GNU public license with its current version 2.41, released in December 2003. Greenstone used C++as the main programming language for supporting modules, Perl as a binding tool to integrate different modules and plug-ins, and Java as a cross-OS tool for the GUI interface. It can run on Windows, UNIX/Linux, Solaris, and MAC OS/X.

(1) *Preservation*. A file loses its original name after the submission process in Greenstone, but keeps its other identities such as size and created date. This is a problem that Greenstone needs to fix in future releases. Moreover, Greenstone does not check data integrity via general checksum techniques, which may not guarantee that an imported digital object is the same as the original one. Regarding the strategy for long-term preservation, Greenstone has some features for long-term preservation as its multiple plug-ins automatically convert files in common formats (e.g. Word, PDF, PS) to their corresponding HTML documents and keep the files in the original formats at the same time.

(2) *Metadata*. Greenstone is very flexible in its metadata support. It can support any metadata sets if the desired metadata schema for the metadata set is provided. The original package includes Dublin Core metadata schema and Greenstone's metadata schema.

(3) *Access*. In terms of external access, Greenstone supports OAI-PMH 2.0 access as a data provider, and a built-in Yaz module which can be used as a Z39.50 client. The use of non-standard persistent URL allows users to access a digital object, but is unable to resist an object's changes in location and state. Greenstone was not rated well on authorization issues, as it only defines three kinds of users: general users, collection builders and administrators, which are not enough to meet our needs for the management of the content and the system.

(4) *System features*. Greenstone provides simple and painless processes for installation and collection building. A single installation file packs all modules, plug-ins, a search engine and a web server together. Greenstone's easy installation process, which is sometimes difficult to find among open source systems, is just as simple as that of commercial packages requiring only a few mouse clicks. In addition, it only takes five straight-forward steps to build a simple collection within a few minutes. Greenstone's extensible search module supports wildcard, Boolean and full-text searches. Search results can be displayed by author, subject and collection. Greenstone not only has a built-in Unicode support for multiple-language collections, but also provides multiple-language user interfaces such as Russian, German, Spanish, and multiple Asian Languages. As a result, Greenstone is popular in European and Asian countries.

(5) *Other*. Greenstone is a fairly mature system and it scored highly against our requirements. However, various programming languages used in Greenstone and the lack of such skill sets in our team result in its unsuitability to be our desired system based on our feasibility analysis. The weakest point of Greenstone is that it does not have a built-in workflow process. Users are unable to customize it to meet their unique workflow needs while building various collections. In addition, the programming languages used by Greenstone

Library Hi Tech
Volume 22 · Number 4 · 2004 · 355-365

are C++, Perl, and Java, and therefore implementing Greenstone will require our technicians to gain extensive programming experience. Our team lacks the required skill sets, lowering its score in the feasibility analysis.

DSpace

DSpace [5] is digital institutional repository software designed to preserve, index and redistribute the intellectual output of an organization in digital formats. The underlying philosophy of DSpace is that the information that the system deals with will outlive the system itself (Bass *et al.*, 2002a, p. 1). Developed jointly by MIT Libraries and Hewlett-Packard (HP), DSpace is a five-year collaboration project started at 2000. DSpace Federation is also scheduled to be established to include all research institutions using DSpace dedicated to the mutual benefits of DSpace development and members' needs. The package, originally released in November 2002, may add some important features in the future releases. It is built in Java and server-side Java technologies, including Java Servlets, JSP, Taglets, Filters, Java Bean Activation Framework, and Java Mail. DSpace uses the PostgreSQL database as its default backend database to store user and system information and metadata records. It can also be customized to use other databases such as Oracle.

(1) *Presentation*. DSpace is the best among these candidates as its theory is to make the content outlast the system. DSpace keeps a file's original name, size and created date. In addition, we are particularly interested in its built-in data integrity check by using MD5 (a "message digest" algorithm for security applications) to ensure the correctness of each file. More importantly, it defines a migration strategy including introducing the concept of file formats as a hierarchy of "unknown", "known", and "supported". DSpace's consideration for scalable storage allows the system to use multiple hard drives, which is particularly useful for an institutional repository, although this feature is limited by the capability of a server's operating system.

(2) *Metadata*. DSpace has limited support for metadata, as its metadata approach is to associate each item with one Dublin Core metadata record. This design consideration results in the limitation of using metadata, since only Dublin Core metadata records are allowed, and one-to-one relationship between item and metadata is too restricted. Theoretically, DSpace states that it will support multiple domain-specific metadata sets (Bass *et al.*, 2002b, p. 3), but at this time only Dublin Core is supported. Another research project called SIMILE [6] (MIT, 2003) at MIT is currently investing to support these domain-specific metadata sets, but its production code might not be available for one to two years. Our preference for the relationship between items and metadata is one-to-many, since an item may require multiple metadata records to meet different users' need. For instance, a manuscript can have its MARC metadata for our library catalogue and at the same time it can also have corresponding Dublin Core metadata for other uses. Furthermore, there is no way to associate metadata with bitstreams, a series of relevant bits of a file. For instance, an item can have a PDF and a HTML bitstreams, and each bitstream might need its own associated metadata due to different nature.

(3) *Access*. The implementation of a CNRI (Corporation for National Research Initiatives) handle system in DSpace advances an additional step above the concept of persistent URL, which allows an object to persist over its changes in location/system and to be accessed in the future. DSpace supports OAI-PMH 2.0 access, but is not Z39.50 complaint, which is a desirable feature for our system. We have already customized DSpace's source code to integrate the system with the University of Arizona LDAP authentication system. DSpace's authorization meets almost all of our requirements, since it defines multiple kinds of users: general users, contributors, content developers and administrators and flexible assignment of groups.

(4) *System features*. The installation process of DSpace is not as simple as that of Greenstone. Due to license issues of multiple open source packages utilized in DSpace, DSpace is unable to pack them together into one zipped file. Users have to download these packages and customize them individually, which requires understanding of the structure of DSpace, related software and UNIX. Our experience shows that it takes some efforts to get it running. DSpace provides basic searching by using the open source search engine Lucene, which allows users to do wildcard and Boolean searches. Full-text search is not supported currently, but can be implemented with customization.

DSpace arranges digital objects into its own predefined three-level hierarchy: communities, collections, and items. A community is the highest level organizational bodies in an institution such as departments, which can contain one or many collections. A

Library Hi Tech
Volume 22 · Number 4 · 2004 · 355-365

collection, the second highest level, is a group of logically related materials that can contain one or many items. An item is an "archival atom" which can contain one or many bitstreams. A bitstream stores the content of an item and metadata, and is organized into bundles with other tied bitstreams (Bass *et al.*, 2002a, p. 3). DSpace developers realized the flexibility of communities, collections, items, bundles, and bitstreams and the relationship between them are all many-to-many. The three-level structure of communities, collections, and items, however, limits the ability to build multiple-level hierarchical structures for some collections. Too often an archival collection can have more than a three-level hierarchy, which might not be feasible to be supported with the current structure. Currently users also find that it is problematic to store HTML documents including a HTML file, its associated CSS (Cascading Style Sheets) and images. The reasons are: first, for each file of the HTML documents (the HTML file, CSS, and image files) one has to go through all submission steps to submit, which is tedious in the collection building process. Second, there is a problem when users view the stored HTML documents. Within the system, users can only view a single file at a time and the result will not be the same as the original HTML documents displayed in browsers such as Internet Explorer, simply because the relationships between these files are gone. An alternative approach is to store a zipped version of these files, but this has obvious drawbacks. Users have to download the zipped file and they must also have the appropriate software to unzip the file. This approach creates an additional unnecessary layer on the top of a complex system.

(5) *Other.* DSpace received the highest score in our feasibility analysis when matched against our requirements, although it is not totally a mature product. Among the three systems, DSpace has the most extensive documentation, ranging from the system architecture to developer resource.

DSpace is the only candidate that supports configurable workflow process and submission policy. The workflow is integrated with the system's built-in authorization to enhance the flexibility and the security of the system. ThesesAlive [7] proposed by five British Universities utilized the configurable workflow and customized DSpace for their Electronic Theses and Dissertations project. As DSpace is implemented with Java and Java's web technologies, our team currently has its required skill sets.

Fedora

The Fedora project [8] was funded by the Andrew W. Mellon Foundation to build a digital object repository management system based on the research paper, "Flexible Extensible Digital Object and Repository Architecture (FEDORA)". Fedora is developed jointly by University of Virginia and Cornell University. The system, designed to be a foundation upon which interoperable web-based digital libraries, institutional repositories and other information management systems can be built, demonstrates how distributed digital library architecture can be deployed using web-based technologies such as XML and Web Services. Fedora, initially shipped with McKoi, a RDBMS written in Java, can be configured using other databases such as Oracle. Fedora, distributed under Mozilla public license, was original released in May 2003.

(1) *Preservation.* Fedora provides some consideration for preservation issues, since it allows multiple versions for a digital object. This versioning feature is hard to find in other content management candidates, but is desirable for digital content management. The system neither checks data integrity nor describes the migration strategy.

(2) *Metadata.* Fedora extends its own version of METS in order to flexibly manage digital objects within a repository and to exchange them among repositories (Fedora, 2002, pp. 7, 15). Various metadata sets are supported in Fedora, including Dublin Core and MARC. From a user's perspective, the current system does not provide a simple way to deposit digital objects. The user must create a Fedora-compatible XML document containing references to digital objects to digest them.

(3) *Access.* Similar to Greenstone and DSpace, Fedora supports OAI-PMH, but is not Z39.50 compliant. Fedora currently only supports persistent URL for access to the content, and does not consider any methods to deal with an object's changes in the future. The authorization in Fedora is pretty much the same as that of Greenstone, which is not sufficient for us to manage the content and the system. In addition, Fedora has built-in access restriction features, which restricts access based on machines' IP addresses. The granularity of current access restriction is system-wide, which is not really applicable for our environment, since it is easy to set up a packet filtering layer firewall such as IPTables that demonstrates much more powerful features. Fedora's Phase II will look for advanced Access Control and authentication, which may bring more useful and meaningful ways to fulfill users' needs.

Library Hi Tech
Volume 22 · Number 4 · 2004 · 355-365

(4) *System features*. Fedora's installation process is not difficult. Its search engine provides wildcard and Boolean searches, but no full-text search capability. However, we are not satisfied with the searching quality. The immaturity of its built-in search engine raised our concerns, as a record may not be found with this search engine. Besides its general administrator and client interfaces, Fedora provides unique access and management services through web Services, which use HTTP as a transport protocol to administer and access digital objects. The separation of its management Application Programming Interface (API) and Access API makes the system easy to manage. It also provides public API to allow users to access and customize the system.

(5) *Other*. The approach that Fedora took is very different from those of Greenstone and DSpace. There is only a concept of digital objects in Fedora. A digital object is a data structure with a Persistent Identifier, one or more datastreams (data and metadata), and one or more disseminators. A disseminator defines an object's datastreams' presentation and behaviors. Similar to DSpace's implementation, Fedora was programmed in Java and Java technologies, which are in our team's skill sets. Therefore, Fedora received high scores in the technical and economic sections of our feasibility analysis. Fedora utilizes extended Metadata Transmission and Encoding Standards (METS) to store information about digital objects, which allows for a high degree of flexibility with respect to adding additional Metadata schemas or multiple versions of an object. The unique features, such as digital objects versioning and Web Services, make it flexible for content management. With these nice features users can build up their desirable and powerful systems, but require a lot of programming and customization. It is, unfortunately, not a ready-to-use CMS based on our environment.

Based on our feasibility analysis (see Appendix 2), DSpace received the highest marks in operational analysis, schedule analysis and economic analysis, while Fedora received the highest score in technical analysis. The overall scores show that DSpace was ranked first among these systems and was chosen as our desired CMS.

Summary

The increasing demands for access to digital projects reveal the need for a digital content management system. We took the system analysis approach that allowed us to identify the problems, draft functional and non-functional requirements, and did feasibility analysis to evaluate commercial and open source systems to meet our needs. We have a clear understanding of the content management industry and different systems in business and technology perspectives. The understanding of advantages and pitfalls of each system helps us polish our strategy to efficiently manage digital content in a cost-effective manner.

Notes

1 http://digital.library.arizona.edu/cowpuncher/
2 www.library.arizona.edu/gifs/ads/ads.htm
3 www.steptwo.com.au/cm/cms/index.html
4 www.greenstone.org
5 www.DSpace.org
6 http://web.mit.edu/simile/www
7 www.thesealive.ac.uk
8 www.fedora.info

References

Bass, M.J., Branschofsky, M., Stuve, D., Breton, P., Tansley, R., Carmichael, P., Cattey, B., Chudnov, D. and Ng, J. (2002a), *DSpace – Technology & Architecture*, MIT, Cambridge, MA, available at: www.DSpace.org/technology/architecture.pdf (accessed 1 October 2003).

Bass, M.J., Branschofsky, M., Stuve, D., Breton, P., Tansley, R., Carmichael, P., Cattey, B., Chudnov, D. and Ng, J. (2002b), *DSpace – Functionality*, MIT, Cambridge, MA, available at: http://DSpace.org/technology/functionality.pdf (accessed 10 Oct 2003).

Boiko, B. (2002), *Content Management Bible*, Hungry Minds, New York, NY.

Fedora (2002), *Mellon Fedora Technical Specification Version 1.1*, University of Virginia, Charlottesville, VA and Cornell University, Ithaca, NY, available at: www.fedora.info/documents/master-spec-12.20.02.pdf (accessed 10 October 2003).

Mandelbaum, J.B. (1992), *Small Project Automation for Libraries and Information Center*, Meckler, Westport, CT.

Microsoft (2002), *Microsoft Computer Dictionary*, 5th ed., Microsoft Press, Redmond, WA.

MIT (2003), "Semantic interoperability of metadata and information in unlike environments", MIT, Cambridge, MA, available at: http://web.mit.edu/simile/www/ (accessed 26 November 2003).

Walton, R.A. and Taylor, N. (1986), *Directory of Microcomputer Software for Libraries*, The Oryx Press, Phoenix, AZ.

Whitten, J.L. (2001), *Systems Analysis and Design Methods*, 5th ed., McGraw-Hill, Boston, MA.

Further reading

Witten, I.H. and Boddie, S. (2003), *Greenstone Digital Library Installer's Guide*, Department of Computer Science, University of Waikato, Hamilton, available at: http://prdownloads.sourceforge.net/greenstone/Install-2.39-en.pdf (accessed 10 Oct 2003).

Library Hi Tech
Volume 22 · Number 4 · 2004 · 355-365

Appendix 1

Table AI The content management system requirements with systems comparison

Functional Requirements	Greenston	Fedora	Dspace
1. Organization Requirements			
1.1 Metadata			
1.1.1 Each digital object must have one or more associated metadata records. (see 1.1.5)	✓	✓	✓
1.1.2 Users must be able to create metadata records.	✓	✓	✓
1.1.3 Users must be able to modify metadata records.	✓	✓	✓
1.1.4 The system must allow for metadata extensibility and complexity.	✓	✓	✓
1.1.5 The system must allow users to associate metadata with digital objects (a many-to-one relationship)	✓	✓	X
1.1.6 The system should support metadata versioning.	X	X	X
1.1.7 The system must support different metadata schemes for collections/sub-groups/etc.	✓	X	X
1.1.8 The system should support different metadata Schemas (e.g. Dublin Core, EAD etc.) for the same collection/sub-group/etc.	✓	✓	\X
1.1.9 The metadata schema must provide rights information field.	✓	✓	✓
1.2 Content			
1.2.1 Users must be able to upload new digital objects.	✓	✓	✓
1.2.2 Users must be able to download (export) digital objects and all associated metadata from the system.	✓	✓	✓
1.2.3 Users must be able to modify (add/delete/update)digital objects (simple objects or compound objects) within the system.	✓	✓	✓
1.2.4 The system should support digital objects versioning.	X	X	X
1.2.5 The system must allow digital objects to be grouped into organized structure (i.e. linear/hierarchical/graphical structure).	\X	X	✓
1.2.6 The system must allow digital objects to be grouped into classification/subjects.	✓	X	✓
1.2.7 The system should allow a digital object to be a member of multiple collections.	\X	X	\X
1.2.8 The system should allow a collection to be a member of multiple collections.	✓	X	X
1.3 Other			
1.3.1 The system must provide online help.	✓	✓	✓
1.3.2 The system help must be accessible all the time.	✓	✓	✓
1.3.3 The system must provide a persistent, unique URL for a digital object.	✓	✓	✓
1.3.4 The system must export content and metadata as the original format.	✓	✓	✓
1.3.5 The system should be able to index digital objects when appropriate.	✓	✓	✓
1.3.6 The system should be able to configure workflow processes.	X	X	✓
2. Presentation Requirements			
2.1 It is desirable to transform one Metadata schema to another Metadata schema (e.g. Dublin core>>Marc).	X	✓	X
2.2 The system must provide a web-based interface for all its functionalities.	\X	\X	✓
2.3 The system should be able to present a unicode-compatible interface.	✓	✓	✓
2.4 The system should be able to allow users to submit feedback on system problems.	\X	\X	\X
2.5 The system must be able to watermark/banding/branding digital objects when appropriate.	X	X	X
2.6 The system must be able to notify users current state of the task.	✓	\X	✓
2.7 The system must be able to provide statistical usage and relevant reports.	X	\X	\X
3. Access Requirements			
3.1 Internal Accessibility			
3.1.1 Users must be able to search content via metadata.	✓	✓	✓
3.1.2 Users should be able to browse digital objects.	✓	✓	✓
3.1.3 The system should provide full-text searching.	✓	\X	\X
3.1.4 The system should provide wildcard searching.	✓	✓	✓
3.1.5 The system should provide Boolcan (AND, OR, NOT)searching.	✓	✓	✓
3.1.6 The system must display rights information to users.	✓	✓	✓
3.2 External Accessibility			
3.2.1 The system must allow users to add new modules to support desired features via access methods (e.g. Public APIs, web servi	\X	✓	\X
3.2.2 The system must provide help documentation for access methods(APIs etc.)	✓	✓	✓
3.2.3 The system must provide Z39.50 access.	X	X	X
3.2.4 The system must provide OAI-PMH access	✓	✓	✓
3.2.5 The system should provide other standard web services access to content and metadata.	X	✓	X
3.3 Authentication and Authorization			
3.3.1 The system must be able to support different roles.	\XX	\XX	✓
3.3.2 The system must be able to support the feature that groups are assigned to one or more roles.	\X	X	✓
3.3.3 The system must be able to support the feature that users are assigned to one or more groups.	✓	\X	✓
3.3.4 The system should support the following roles: Min Role, General User, Contributor, Content Developer, Administrator, Max Role			
3.3.4.1 Min Role: no access at all (or other desired access points).	X	X	X
3.3.4.2 General User Role: only can browse and search metadata and content, in addition to Min Role's role	X	✓	✓
3.3.4.3 Contributor Role: must be able to submit metadata and content, in addition to general users' role.	X	X	✓
3.3.4.4 Content Developer role: must be able to create/edit/add/delete metadata and content in defined(not ALL) collections, in additi	✓	X	✓
3.3.4.5 Administrator role: must be able to create/add/delete users and groups.	✓	✓	✓

(*continued*)

Table AI

3.3.4.6 Max Role: must be able to do ANYTHING within the system.	X	X	X
3.3.5 The system must use authorization to support different roles within this system	✓	✓	✓
3.3.6 The system must use authentication to verify users with campus security system (NetID).	X	\X	✓
3.4 Systems			
3.4.1 The system must support remote and multiple access for user with different roles(i.e. client-server access model).	✓	X	✓
3.4.2 The system must be ACID (Atomicity, Consistency, Isolation, Durability).	✓	✓	✓
3.4.3 The system must be unicode compatible for all functionalities	✓	✓	✓
4. Preservation Requirements			
4.1 The system must store metadata records separately from content.	✓	✓	✓
4.2 The system must be able to store any file format.	✓	✓	✓
4.3 The system should support a set of known file types more inherently (MS Word, for example)	✓	X	✓
4.4 The system should have version control for metadata and content.	X	X	X
4.5 The system must be able to keep error/usage/debug log files.	✓	✓	✓
4.6 The system must keep the original file's name, size and created date.	X	X	✓
4.7 The system must handle scalable storage.	\X	\X	✓
Non-functional Requirements			
1. Performance			
N1.1 The system should be scalable in terms of storage and upgrade.			
N1.2 The system must notify and update users' request status within reasonable time.			
2. Information			
N2.1 The system must ensure the accuracy of metadata and content.			
(a) By creating the advisory group			
(b) By enforcing file checksum for both submission and download.			
(c) By enforcing submission policies.			
3. Economy			
N3.1 The system (hardware and software) must be cost effective.			
N3.2 The system maintenance (staff, hardware and software) must be included in cost assessment.			
N3.3 We must measure the cost of the skill sets within the library against that of obtaining the lacked skill sets.			
N3.4 The system should be provided continuous support from the vendor or groups (considering the community)			
4. Control			
N4.1 Content must adheres collection development policies.			
N4.2 Workflow policies/processes for born digital/digitized content should be well documented.			
N4.3 Files must be able to be backuped and must be off-site storage.			

Source: University of Arizona Library Updated (November 2003)

Library Hi Tech
Volume 22 · Number 4 · 2004 · 355-365

Appendix 2

Table AII Feasibility analysis based on our environment

The feasibility analysis is based on the following factors and the weight of each factor is assigned based on our environment.

1. **Operational** (how well will it work for our environment? How well will people feel about it?).

Weight: 50%,
a) Functionality (calculated from Appendix A "Functional Requirements" section) 50%
b) Politics (from users, management people views) 0%

2. **Technical** (technical solution maturity, team available skills and resources)

Weight: 20%, each 10%
a) Technology (assessment of the solution maturity, extensibility) 10%: 10 (very mature and extensible); 1 (very immature and restricted).
b) Expertise (assessment of team member skills (7%), setup (1%), customization(1%), management (1%))

3. **Schedule** (how long will the solution take to implement?)

Weight: 10%
a) schedule 10%: 10 (within a week); 1 (more than 1 year)

4. **Economic** (software/hardware cost, development cost, maintenance cost, etc.)

Weight: 20%, each 5%.
a) Software cost (5%): 5 (open source software), reducing marks for commercial software
b) Hardware cost (5%): 5 (general platforms); reducing marks for specialized platforms.
c) Development cost (5%): 5 (no customization); 1 (80% more customization)
d) Maintenance cost (5%): 5 (required skills are in our team's skills set); 1 (80% skills must be learned).

Feasibility Analysis		Greenstone	Fedora	DSpace
Operational (50%)				
Functionality	50%	33.9	29.4	35.75
Politics	0%			
Technical (20%)		11.67	18.5	18.33
Technology	10%	6.33	8.67	8.67
Expertise	10%	5.33	9.83	9.67
Schedule (10%)	10%	5.67	4.33	8.67
Economic (20%)		15.5	17	18.67
Software cost	5%	5	5	5
Hardware cost	5%	5	5	5
Development cost	5%	2.67	3	4.67
Maintenance cost	5%	2.83	4	4
Total	**100%**	**66.74**	**69.23**	**81.42**