Refereed article received
26 January 2006
Revision approved for
publication 13 March 2006

A checklist for evaluating open source digital library software

Dion Hoe-Lian Goh, Alton Chua, Davina Anqi Khoo, Emily Boon-Hui Khoo, Eric Bok-Tong Mak and Maple Wen-Min Ng

Division of Information Studies, School of Communication and Information, Nanyang Technological University, Singapore

Abstract

Purpose – Many open source software packages are available for organizations and individuals to create digital libraries (DLs). However, a simple to use instrument to evaluate these DL software packages does not exist. The objectives of the present work are to develop a checklist for DL evaluation and use this checklist on four DL software packages.

Design/methodology/approach – Features that characterized "good" open source DL software were determined from the literature. First identified were essential categories of features that DL software should possess. These categories were then decomposed into supporting features. From these, a checklist that covered all such features was developed. The checklist was then used to evaluate four popular open source DL software packages (CDSware, EPrints, Fedora, and Greenstone) for the purposes of assessing suitability for use in a DL project to be undertaken by the authors.

Findings – A checklist consisting of 12 categories of items was developed. Using this, Greenstone was found to be the best performer, followed by CDSware, Fedora and EPrints. Greenstone was the only software package that consistently fulfilled the majority of the criteria in many of the checklist categories. In contrast, EPrints was the worst performer due to its poor support for certain features deemed important in our checklist, and a total absence of functionality in other categories.

Originality/value – The present work attempts to develop a comprehensive checklist for assessing DLs. Its flexibility allows users to tailor it to accommodate new categories, items and weighting schemes to reflect the needs of different DL implementations.

Keywords Digital libraries, Program testing, Computer software

Paper type Research paper

Introduction

Digital libraries (DLs) facilitate creation, organization and management of multimedia digital content and collections, and provide search, retrieval and other information services over computer networks and other electronic media. Developments in DL technologies have changed the way people access and interact with information, and have also extended the concept of libraries far beyond physical boundaries.

Digital library systems have the potential to empower users, not just librarians, to conceive, assemble, build and disseminate new information collections (Bainbridge *et al.*, 2003). Therefore, one of the key functionalities of a DL should be the matching of user work patterns. To achieve this, a thorough understanding of the users of libraries and the system itself should be obtained. Apart from the need for deeper understanding of users, the fit between the tools used to craft the DL and the necessary requirements has to be ascertained.

Online Information Review
Vol. 30 No. 4, 2006
pp. 360-379

1468-4527
DOI 10.1108/14684520610686283

The Asian Communication Resource Center (ACRC), located in Nanyang Technological University (Singapore), is a regional center dedicated to developing information resources on different aspects of communication, information, media and information technology disciplines, especially in Asia. The center has about 20 years' worth of documents including books, journals, conference proceedings and working papers. To make these materials more accessible to users who are not based in Singapore, the ACRC decided to explore the possibility of building a DL to provide online access to these materials.

It was decided in the early stages of the project that open source DL software was to be used, given the amount of work done in this area by academics and practitioners in the field, as well as the opportunities for using the DL as a research platform to test new concepts and technologies. However, a search of the literature yielded little in the evaluation of DL software, confirming Saracevic's (2000) observation that much effort has been put into DL research and practice but not so much on evaluation. This gap led to the present work. Specifically, we conducted a study that examined the features of four DL software packages against a set of predetermined criteria that were deemed to be essential for the development of the ACRC DL. Our objectives were to:

- Determine the features that characterize "good" open source DL software and develop a checklist from these features. Here, we identified essential categories of features DL software should possess. Examples include content management, searching, and metadata standards. These categories were then decomposed into supporting features. From these, a checklist that covered all such features was developed.
- Evaluate the features of candidate open source DL software against the checklist. The checklist was applied to four open source DL software packages that were identified as possible candidates for the ACRC DL. Scores were assigned to each DL depending on its level of support for the features in the checklist.
- Analyze the strengths and weaknesses of the DL software. After scoring, a comparison of the features across the DL software packages was examined with the objective of identifying similarities, differences, strengths and weakness of our candidate software.

The remainder of this paper is structured as follows. In the next section, open source concepts and a description of our candidate open source DL software are presented. The development of our checklist and the evaluation criteria used are discussed. This is followed by a presentation and analysis of the evaluation results of the DL software packages. We conclude with opportunities for future work and a general discussion of our evaluation experience.

Open source software

Open source software has been a nebulous reference to any software that is free, and is often confused with freeware and shareware. The Open Source Initiative (OSI; www.opensource.org) has therefore become a certification body for open source software under a commonly agreed-upon definition for "open source". Highlights of the OSI's

definition of open source include: free distribution and redistribution of software and source code; licenses that allow distribution of modifications and derived works and non-discrimination against persons, groups or fields of endeavor (Open Source Initiative, 2005). In contrast to open source, freeware refers to software that is released free of cost in binary format only, and its licenses usually prohibit modifications and commercial redistribution. On the other hand, shareware is software that is released free of cost in binary format but only for a limited trial period after which users are encouraged to purchase the software.

The availability of the source code in open source software allows users to modify and make improvements to it, and such contributions could come from a diverse talent pool of programmers. Thus, open source software tends to have more functions, being developed by the users of the software themselves, as compared to commercial software, where a vendor's priority is in profit generation that may not be inline with the needs of users (Von Hippel and von Krogh, 2003). Further, because source code is accessible and modifiable, contributions also lead to improvements in the functionality of the software (Morgan, 2002). In addition, updates can be obtained at low or even no cost and there are no royalties or license fees. There is less likelihood of being dependent on a single software provider or being trapped into long-term software support contracts, which restricts flexibility in implementation (Surman and Diceman, 2004).

However open source software has its disadvantages. One of the more common complaints is the lack of formal support and training that a commercial software package would offer (Caton, 2004). Often, support is provided through mailing lists and discussion forums. In addition, open source software is also not known for ease of use as the focus is usually on functionality. Consequently, open source adopters will have to take greater personal responsibility in terms of leveraging staff expertise to implement and maintain their systems, including hardware and network infrastructure (Poynder, 2001).

Nevertheless, open source is increasingly considered as an alternative to commercial digital library systems due to dissatisfaction with functionality (Breeding, 2002). Another factor is the increasing budget cuts that libraries face (Evans, 2005). The cost of software production and maintenance is also rising dramatically. As a result, open source DL software, with its free access and good level of functionality, are some reasons for increased usage and interest.

Digital library software selection and evaluation

While there exists a variety of open source DL software available for use, it was decided that the ACRC DL be deployed on software that was stable, standards-based and had a reasonable number of installed bases. The criteria for initial selection of DL software therefore included the following:

- The software must be available for download and installation at no cost via an open source license to facilitate evaluation.

- The software should be relatively well known and commonly used, and this was inferred from the number of bases installed, especially in credible organizations such as universities.
- The software must be supported either on Linux or Windows, as these are commonly used platforms.

Given these criteria, four DL packages were selected for our evaluation – CERN document server (CDSware), EPrints, Greenstone and Fedora:

- *CDSware*. CERN, the European Organization for Nuclear Research, is the world's largest particle physics laboratory. CERN developed the CERN document server software, also known as CDSware (http://cdsware.cern.ch), to manage its collection of information. This includes over 800,000 bibliographic records and 360,000 full-text documents including preprints, journal articles, books and photographs. The software is released under the GNU open source license.
- *EPrints*. EPrints (www.eprints.org) was developed at the University of Southampton, with the first version of the software publicly released in late 2000. The objective behind the creation of EPrints was to facilitate open access to peer-reviewed research and scholarly literature through OAI. However, EPrints also serves as an archive for other electronic documents such as images and audio. EPrints currently has an installed base of more than 190.
- *Fedora*. Fedora (www.fedora.info) is jointly developed by the University of Virginia and Cornell University, with its first version released in 2003. The objective of Fedora 1.0 was to create a production quality system using XML and web services to deliver digital content. Fedora supports digital asset management, institutional repositories, digital archives, content management systems, scholarly publishing enterprises and digital libraries. The system is designed to be a foundation upon which full-featured institutional repositories and other interoperable web-based digital libraries can be built. It currently has a distributed installed base of more than 360, with collection sizes of 10 million objects.
- *Greenstone*. Greenstone (www.greenstone.org) is developed by the New Zealand Digital Library Project at the University of Waikato, and distributed in cooperation with UNESCO and the Human Info NGO. Greenstone is a tool for building libraries and aims to empower users, particularly in universities, libraries and other public service institutions to build large distributed digital library collections. The current installed base is unknown but the number of downloads of the software appear to be large.

Software evaluation with checklists

To effectively evaluate the chosen DL software, a framework is necessary to guide the planning, controlling and reporting of the evaluation of software products. Common elements between these software packages need to be examined so that suitable

conclusions can be drawn. To accomplish this, evaluation instruments (Punter, 1997) are needed, and several types are available including:

(1) static analysis of code, for structural measurement or anomaly checking;
(2) dynamic analysis of code, for test coverage or failure data;
(3) reference tools that compare the software product;
(4) reference statistical data; and
(5) inspection with checklists.

Although the first three evaluation methods are usually looked upon as well founded and applicable to software evaluation, experience shows that the use of checklists is necessary (Punter, 1997). Checklists have been used widely to verify the correctness of software documentation and software code. Gilb and Graham (1994) defined them as "a specialized set of questions designed to help checkers find more defects"; whereas another largely accepted definition is that a checklist is a "list of questions that clarify and standardize the inspection process and guide inspectors to pay attention to the right things" (Tervonen and Kerola, 1998).

Evaluation criteria

In designing a DL, there is no decision more important than the selection of quality software that forms the platform from which services are delivered. The variety of choices however makes the selection somewhat daunting, and the key is a careful definition of the nature of the information in the library and how it will be used (Dobson and Ernest, 2000). In the present work, a discussion with ACRC staff and potential users, coupled with a review of the DL literature, yielded the following five broad requirements that were used as our evaluation criteria:

(1) *Content management*. This requirement is related to the ease with which content is created, submitted, reviewed, organized and versioned within the DL. It also encompasses the provision for searching and browsing functions such as metadata search, full-text search, and hierarchical subject browsing. Additionally, content encoded in a variety of popular formats including text (e.g. ASCII, UNICODE, RTF), image (e.g. TIFF, GIF, JPEG), presentation (e.g. Adobe PostScript and Adobe PDF), structured formats (e.g. HTML and XML), audio and video (e.g. Real, MP3, AVI and MPEG) ought to be supported.
(2) *User interface*. The user interface requirement covers the flexibility in customizing the interface to suit the needs of different digital library implementations as well as the support for multilingual access. With multilingual access, the user is able to specify the language for the DL's user interface as well as the cataloging information stored within it (Witten and Bainbridge, 2002).
(3) *User administration*. This requirement concerns the range of functions needed to manage the users of the DL. For example, access to content in the DL needs to be restricted through password authentication, IP filtering, and proxy filtering. Also, usage patterns have to be monitored and reported. When usage patterns

are analyzed, the needs and interests of DL users can be better understood (Jones *et al.*, 1998).

(4) *System administration*. This requirement is related to the back-end maintenance of the DL. Automatic tools are useful particularly for large DLs where maintenance work is labor-intensive (Arms, 2000). Functions such as automated content acquisition, harvesting and automatic metadata generation, including named entity recognition and automatic subject indexing/classification, make DL maintenance much easier. Next, the DL needs to support preservation standards as well as persistent document identification, so that the transfer of digital materials from one hardware/software configuration to another would not compromise reference citations and other links (Cordeiro, 2004; Hedstrom, 2001).

(5) *Other requirements*. The DL needs to be interoperable with other systems to which it is connected. This allows each system to evolve independently without sacrificing their ability to communicate with each other (Paepcke *et al.*, 2000). At least two basic interoperability protocols should be supported, namely, Z39.50 and OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting). Additionally, the DL must be compliant with standards established for digital library collection and services. Examples of standards include XML for representation of information; XHTML for web pages; GIF, TIFF and JPEG for images; Unicode for multilingual support and information interchange; and Dublin Core and MARC 21 for metadata. Finally, the DL needs to provide mechanisms through which DL administrators and developers can obtain system support and help. Such mechanisms include documentation, manuals, mailing lists, discussion forums, bug tracking, feature request systems and formal helpdesk support.

The DL evaluation checklist

Due to the lack of a universally accepted definition for a digital library, there has not yet been a common methodology for the selection of good digital library software. With this in mind, the present study aimed to develop a simple-to-use instrument for evaluating DL software with the following characteristics:

- *Comprehensiveness*. The evaluation criteria should cover all the key areas involved in DL software selection.
- *Usability*. The instrument should be simple to understand, and more importantly, easy to employ by a variety of users with or without a background in software evaluation.
- *Flexibility*. In choosing DL software, different stakeholders may place different emphasis on the various criteria. For example, a library administrator may be more concerned with the ease of submission of material, whereas a researcher may be more interested in better search functionality. The weights associated with each criterion should be easily modifiable to reflect different stakeholder needs.

- *Expandability*. The instrument should be easily expandable to include additional factors and criteria such as new and emerging standards.

Our DL evaluation checklist consists of 12 categories of items, each with varying degrees of importance: content management, content acquisition, metadata, search, access control and security, report and inquiry, preservation, interoperability, user interface, standards compliance, automatic tools and support (the checklist is found in the Appendix).

Methodology

The method of assigning weights to evaluation criteria was adapted from the Edmonds and Urban's (1984) methodology, who recommended the use of the Delphi technique. In the original technique, a committee anonymously assigns weights to each criterion, usually through a questionnaire. The committee reviews the results and if there is no consensus, these steps are repeated until a consensus is reached. In the present study, we modified the Delphi technique by having a group of four people trained in information science and familiar with DL concepts to assign weights to each category and its respective items independently. The total sum of the category weights was 100, while the total sum of the items in each category was 10. Discrepancies were then resolved with face-to-face discussions in which each person provided justifications for the reasons behind their decisions. Pair-wise comparisons were also conducted as part of the process of formulating appropriate weights for the checklist. In pair-wise comparison, the relative importance of each criterion against every other criterion is determined, often in a group session that is preceded by individual assessment (Koczkodaj *et al.*, 1997).

Next, the four DL software packages selected in this study were evaluated using the checklist. Scoring was done by evaluating the software as a group, using the same four people that developed the checklist, but on a separate occasion. In cases where the evaluators disagreed over whether a particular criterion was met in a software package, a simple majority vote was taken. In cases of a draw, a consensus was arrived through emailing the DL software developers or consulting other sources.

Findings and analyses

In this section we present our findings on how well the four selected DL software packages scored on a ten-point scale for each of the 12 categories in the DL software evaluation checklist. Table I shows the scores for each of the categories for the four DL software evaluated.

Content management

This category (see Appendix, section 1) involves procedures and tools pertaining to the submission of content into the DL as well as the management of the submission process. As shown in Table I (row 1), all DL software, with the exception of Fedora, satisfied most, if not all, of the criteria. Fedora only managed to score 4.50 out of 10. This comparatively poor performance is due mainly to a lack of submission support

No.	Category	Digital library software package: Eprints 2.3.11	Fedora 2.0	Greenstone 2.51	CERN CDSware 0.5.0
1	Content management	8.00	4.50	9.00	10.00[a]
2	Content acquisition	10.00[a]	9.44	9.17	8.67
3	Metadata	5.13	7.00[a]	5.75	6.25
4	Search support	3.40	4.96	7.25	8.71[a]
5	Access control and privacy	8.33[a]	6.17	7.83	7.67
6	Report and inquiry capabilities	0.00	6.00	10.00[a]	0.00
7	Preservation	5.50	10.00[a]	2.50	7.50
8	Interoperability	5.00	6.00	8.00[a]	5.00
9	User interface	10.00[a]	10.00[a]	10.00[a]	10.00[a]
10	Standards compliance	10.00[a]	10.00[a]	10.00[a]	10.00[a]
11	Automatic tools	5.00	5.00	10.00[a]	10.00[a]
12	Support and maintenance	6.00	10.00[a]	10.00[a]	10.00[a]

Note: [a]Indicates the highest category score

Table I.
Evaluation results for individual categories

and review. Fedora only provides capabilities to insert content, but not features such as notification of submission status or allowing users to modify submitted content.

Content acquisition

Content acquisition (see Appendix, section 2) refers to functions related to content import/export, versioning and supported document formats. Table I (row 2) shows that all the DLs managed to fulfill most of the criteria. EPrints in particular achieved a full score of 10.

Metadata

As mentioned earlier, metadata support (see Appendix, section 3) in DLs is vital in content indexing, storage, access and preservation. However, the performance in this area was disappointing. As shown in Table I (row 3), most of the DLs in our study only supported a few metadata standards. While it is encouraging that at least core standards like MARC21 and Dublin Core were supported, emerging metadata schemas such as EAD and METS (Guenther and McCallum, 2003) were neglected by all except Fedora.

Search support

Search support refers to a range of searching and browsing functions such as metadata search, full-text search, and hierarchical subject browsing. Considering the importance of searching functionality (see Appendix, section 4), Table I (row 4) shows that performance across the four DLs was varied, ranging from a low of 3.40 to a high of only 8.71. In particular, EPrints' poor performance was due to the absence of full-text search support as well as metadata search. In addition, none of the software evaluated featured proximity searching capabilities.

Access control and privacy

Access control and privacy include the administration of passwords, as well as the management of users' accounts and rights to specified locations within the DL. Most of the DLs surveyed scored well for this indicator (see Table I, row 5; see Appendix, section 5), with EPrints being the best overall performer. CDSware obtained the best score in password administration, having not only system-assigned passwords, but also the ability to select passwords and to retrieve forgotten passwords. Fedora scored well on access management in its support for IP address filtering, proxy filtering, and credential-based access.

Report and inquiry capabilities

This category is concerned with usage monitoring and reporting (see Appendix, section 6). Table I (row 6) shows that Greenstone was the only software that fulfilled all the requirements in this category. While Fedora provided usage statistics, it did offer report generation tools. Both EPrints and CDSware lacked report and inquiry capabilities.

Preservation

Preservation (see Appendix, section 7) refers to preservation of metadata and quality control measures to ensure integrity, and persistent documentation identification for migration purposes (Hedstrom, 2001). Fedora was a clear winner with its support for CNRI Handles, quality control and provision of a prescribed digital preservation strategy (see Table I, row 7).

Interoperability

Interoperability (see appendix, section 8) is concerned with the benefits of integrating distributed collections and systems (Paepcke *et al.*, 2000). Results indicated that Greenstone was the best performer (see Table I, row 8). All the software surveyed supported OAI-PMH. However, Z39.50 was only supported by Greenstone. This may be due to the protocol being much more complex to implement.

User interface

This category (see Appendix, section 9) deals with support for multilingual access as well as the ability to customize the user interface to suit the needs of different DL implementations. All DL software surveyed obtained a full score (Table I, row 9), reflecting that these issues were taken into consideration.

Standards compliance

Standards are important for the sharing of digital content and long-term digital preservation (Dawson, 2004), and this category (see Appendix, section 10) was thus evaluated by looking for evidence of the usage of standards. As the only other category with a full score across all software (see Table I, row 10), there appears to be a demonstrated commitment to the use of standards. It should be noted however that such a commitment does not imply every conceivable standard should be adopted, and the other evaluation categories should be consulted to determine which specific

standards are supported by each DL. For example, while most document and image format standards are supported by the four DLs, not all metadata formats are, with Dublin Core being the only one supported by all.

Automatic tools

This category refers to tools for automated content acquisition, harvesting and metadata generation (see Appendix, section 11). In the context of DLs, automatic tools are useful for maintenance and can reduce labor costs, especially for large collections (Arms, 2000). Table I (row 11) shows that Greenstone and CDSware obtained full scores while Fedora and EPrints did not fare as well.

Support and maintenance

Support and maintenance (see Appendix, section 12) is an important aspect in all software systems, and open source software is often criticized to be lacking in these. However, our results show that three out of the four DLs evaluated performed well in this category (see Table I, row 12) by offering documentation, manuals, mailing lists, discussion forums, bug tracking, feature request systems, and formal helpdesk support. Only EPrints faired relatively poorly due to its lack of formal helpdesk support, and documentation that was not updated.

Consolidated scores

Figure 1 shows the consolidated scores of the four DL software evaluated. Greenstone emerged as the best performer with a consolidated score of 82.17, followed closely by CDSware with a score of 81.35. This was followed by Fedora and EPrints with scores of 75.01 and 66.49, respectively. Note that the consolidated scores were obtained by

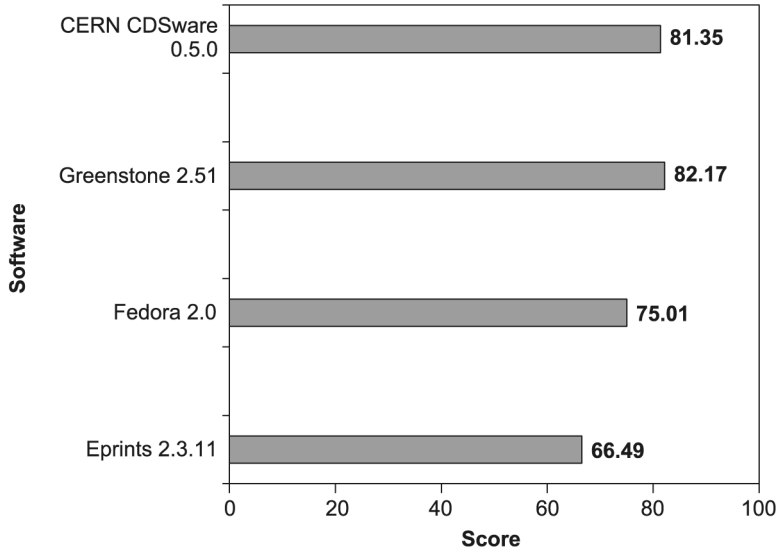

Figure 1.
Consolidated scores for the four digital library software packages

summing all category scores after normalizing by their respective category weights (see the Appendix for details).

Greenstone was the only software package that consistently fulfilled the majority of the criteria in many categories and obtained full scores in five of the 12 categories. These five indicators were report and inquiry, user interface, automatic tools, standards compliance, and support and maintenance. Clearly, Greenstone places great emphasis on end-user functionality. For example, usage reports and statistics help a library administrator to determine bottlenecks and identify popular files accessed. User interface customizability allows different DLs to create interfaces that suit the needs of its stakeholders, while automatic tools simplify content management and acquisition. In addition, Greenstone attained close to perfect scores for content management and acquisition, hence helping to ease the tasks of managing content in the digital library.

In addition, the ease of installation for Greenstone is another winning factor. Packaged in a single executable installation, the DL was operational on a Windows 2003 server machine in less than an hour. Documentation for Greenstone is also extensive. There is a wealth of online documentation and tutorials available on the Greenstone website, and a number of organizations even offer training courses. In sum, we believe that Greenstone is the user-friendliest software for creating digital libraries among the four evaluated.

Similar to Greenstone, CDSware obtained full scores in five of the 12 categories. Although it tracked Greenstone closely in total score, CDSware performed poorer because of its lack of report and inquiry features. This may make system and usage monitoring difficult, an important task as explained earlier. An issue with CDSware not reflected in the checklist was the difficulty in installing the software. CDSware is only available on the Linux platform and requires extensive configuration. As compared to the straight-forward installation of Greenstone, CDSware took several days for installation on a newly setup Linux machine due to the number of other required software packages that needed to be installed and properly configured, and the extensive knowledge of Linux required.

In third place was Fedora, obtaining a full score in four out of the 12 categories. Fedora's key strength is its support for preservation and standards, in which full scores were obtained. It also ranked highest in the metadata category due to its support for many metadata standards. Other than the lack of Z39.50 support, Fedora appears to be a good candidate with regards to long-term digital preservation needs. Fedora was also easily installed on a Windows 2003 server machine, although more configuration work was required when compared to Greenstone. Fedora has limited support for automated tools and content management features.

EPrints was the worst performer with a total score of 66.49. Its advantage was that the software was the only one to obtain a full score for content acquisition and it supports the widest range of document formats. On the other hand, EPrints lacks in usage reporting and inquiry capabilities. It is also only available on the Linux platform and therefore shares the same installation problems facing CDSware. However, its greatest weakness is the low score of 3.40 under the search category. Only metadata searching is supported and full-text search is not available.

Discussion and conclusion

Although the checklist developed in the current study aims to be comprehensive, it represents a first step in the development of an exhaustive evaluation tool and

therefore, the current version has some limitations in the assessment of DL software. For example, the checklist does not take into account factors such as hardware, time, manpower, money and other resources, as these may vary depending on the implementing organization or individual. The availability of application programming interfaces for implementing new features was also not considered. In addition, the weights given to the various categories were assigned through a consensus process among four evaluators. Other stakeholders with different viewpoints and needs may require different weighting schemes. However, as discussed previously, the checklist is flexible enough to accommodate new categories, items and weighting schemes. Saracevic (2000) argues that while resources have been expended on DL research and practice:

> Evaluation is not, by and large, a part of these efforts. With a few notable exceptions in either research or practice, digital library evaluation is not conspicuous . . . digital library evaluation has yet to penetrate research, practice, or even debate.

When this project was first initiated, there was a dearth of information on the evaluation and comparison of DL software. Nevertheless, we wanted to investigate how well our four candidate software packages would perform, with the goals of creating a checklist for DL evaluation and to help guide our recommendations for the ACRC DL itself.

Extensive research was done to extract requirements for DLs. These requirements led to the definition of the criteria for DL evaluation and from these, a checklist was created. Scoring each DL software package against our checklist further reinforced the differences of each DL in accommodating diverse needs. From the results of the evaluation, we have come to the conclusion that current open source DL software still lacks certain functionalities perceived to be important, as gathered from the literature. However, among our four candidate DLs, Greenstone was able to fulfill most of all the vital indicators because of its strong support for end-user functionality. Work is therefore underway to further evaluate Greenstone as a platform for the ACRC DL. CDSware was yet another strong contender, and we believe that it can only get better when it adds a usage monitoring and reporting feature in the next release. In contrast, Fedora and EPrints did not perform as well in the evaluation because they lacked strong support in these areas, and especially in the search category. However, it must be noted that each software package has individual strengths and weaknesses that will appeal to different organizations and stakeholders with different needs. Therefore, those interested in implementing a DL can use the checklist to evaluate how well their candidate software fits into their specific implementation requirements.

References

Arms, W. (2000), "Automated digital libraries: how effectively can computers be used for the skilled tasks of professional librarianship?", *D-Lib Magazine*, Vol. 6 Nos 7/8, available at: www.dlib.org/dlib/july00/arms/07arms.html

Bainbridge, D., Thompson, J. and Witten, I.H. (2003), "Assembling and enriching digital library collections", *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 323-34.

Breeding, M. (2002), "An update on open source ILS", *Information Today*, Vol. 19 No. 9, pp. 42-3.

Caton, M. (2004), "Sugar sale: just the SFA basics", *eWeek*, Vol. 21 No. 50, p. 48.

Cordeiro, M. (2004), "From rescue to long-term maintenance: preservation as a core function in the management of digital assets", *VINE*, Vol. 34 No. 1, pp. 6-16.

Dawson, A. (2004), "Creating metadata that work for digital libraries and Google", *Library Review*, Vol. 53 No. 7, pp. 347-50.

Dobson, C. and Ernest, C. (2000), "Selecting software for the digital library", *EContent*, Vol. 23 No. 6, pp. 27-30.

Edmonds, L.S. and Urban, J.S. (1984), "A method for evaluating front-end life cycle tools", *Proceedings of the 1st IEEE International Conference on Computers and Applications*, pp. 324-31.

Evans, R. (2005), "Delivering sizzling services and solid support with open source software", paper presented at World Library and Information Congress: 71st IFLA General Conference and Council, available at: www.ifla.org/IV/ifla71/papers/122e-Evans.pdf

Gilb, T. and Graham, D. (1994), *Software Inspection*, Addison-Wesley, Wokingham.

Guenther, R. and McCallum, S. (2003), "New metadata standards for digital resources: MODS and METS", *Bulletin of the American Society for Information Science and Technology*, Vol. 29 No. 2, p. 12.

Hedstrom, M. (2001), "Digital preservation: problems and prospects", *Digital Libraries*, Vol. 20, available at: www.dl.ulis.ac.jp/DLjournal/No_20/1-hedstrom/1-hedstrom.html

Jones, S., Cunningham, S.J. and McNab, R. (1998), "Usage analysis of a digital library", *Proceedings of the 3rd ACM Conference on Digital libraries*, pp. 293-4.

Koczkodaj, W., Herman, M. and Orłowski, M. (1997), "Using consistency-driven pairwise comparisons in knowledge-based systems", *Proceedings of the 6th International Conference on Information and Knowledge Management,*, pp. 91-6.

Morgan, E.L. (2002), "Possibilities for open source software in libraries", *Information Technology and Libraries*, Vol. 21 No. 1, pp. 12-15.

Open Source Initiative (2005), *The Open Source Definition*, available at: www.opensource.org/docs/definition.php

Paepcke, A., Brandriff, R., Janee, G., Larson, R., Ludaescher, B., Melink, S. and Raghavan, S. (2000), "Search middleware and the simple digital library interoperability protocol", *D-Lib Magazine*, Vol. 6 No. 3, available at: www.dlib.org/dlib/march00/paepcke/03paepcke.html

Poynder, R. (2001), "The open source movement", *Information Today*, Vol. 18 No. 9, pp. 66-9.

Punter, T. (1997), "Using checklists to evaluate software product quality", *Proceedings of the 8th European Conference on Software Cost Estimation (ESCOM)*, pp. 143-50.

Saracevic, T. (2000), "Digital library evaluation: toward evolution of concepts", *Library Trends*, Vol. 49 No. 2, pp. 350-69.

Surman, M. and Diceman, J. (2004), "Choosing open source: a guide for civil society organizations", available at: www.commons.ca/articles/fulltext.shtml?x=335

Tervonen, I. and Kerola, P. (1998), "Towards deeper co-understanding of software quality", *Information and Software Technology*, Vol. 39 Nos 14/15, pp. 995-1003.

Von Hippel, E. and von Krogh, G. (2003), "Open source software and the 'private-collective' innovation model: issues for organization science", *Organization Science*, Vol. 14 No. 2, pp. 209-23.

Witten, I.H. and Bainbridge, D. (2002), *How to Build a Digital Library*, Morgan Kaufmann, San Francisco, CA.

Appendix. Checklist for digital library evaluation

Figure A1 presents the digital library evaluation checklist. Usage and scoring procedures are as follows:

Table II presents the digital library evaluation checklist. Usage and scoring procedures are as follows:

1. Place a check mark for each item that exists in the software package being evaluated.

2. Within each subcategory,
   a. Multiply the number of check marks by the subcategory weight. This weight also represents the maximum score for that subcategory.
   b. Divide this resulting number by the number of items in the subcategory to obtain the subcategory score.

3. Within each category, sum all subcategory scores, divide the total by 10, and multiply the resulting number by the category weight to obtain the category score.

4. To obtain the consolidated score, sum all category scores.

Table II. Digital library evaluation checklist

	Checklist Categories		Weight	
1.0	**Content Management**			**7.00**
1.1	**Submission Administration**		**3.00**	
	1.1.1 Allows multiple collections within same installation of system	☐		
	1.1.2 Allows repository administrator to set submission parameters	☐		
	1.1.3 Home page for each collection	☐		
	Sub-category score		☐	
1.2	**Submission workflow**		**3.00**	
	1.2.1 Segregated submission workspaces	☐		
	1.2.2 Submission roles	☐		
	1.2.3 Configurable submission roles within collections	☐		
	Sub-category score		☐	
1.3	**Submission support**		**2.00**	
	1.3.1 Email notification for users	☐		
	1.3.2 Email notification for administrators	☐		
	Sub-category score		☐	
1.4	**Submission review**		**2.00**	
	1.4.1 Allows users to review completed content	☐		
	1.4.2 Allows users to review uncompleted content	☐		
	1.4.3 Allows content administrator to review submissions	☐		
	Sub-category score		☐	
	Category Score			☐

(continued)

Figure A1.
Digital library evaluation checklist

	Checklist Categories		Weight	
2.0	**Content Acquisition**			**11.00**
	2.1 **Content import/export**		**3.00**	
	2.1.1 Upload compressed files	☐		
	2.1.2 Upload from existing URL	☐		
	2.1.3 Volume import of objects	☐		
	2.1.4 Volume import of metadata for existing collections	☐		
	2.1.5 Volume export/content portability (to other systems)	☐		
	Sub-category score		☐	
	2.2 **Document/object formats**		**5.00**	
	2.2.1 Administrator ability to limit approved file formats	☐		
	2.2.2 Submitted items can comprise multiple files or file types	☐		
	2.2.3 Text formats	☐		
	2.2.3.1 ASCII	☐		
	2.2.3.2 UNICODE	☐		
	2.2.3.3 RTF	☐		
	2.2.4 Image formats	☐		
	2.2.4.1 TIFF	☐		
	2.2.4.2 GIF	☐		
	2.2.4.3 JPEG	☐		
	2.2.5 Presentation formats	☐		
	2.2.5.1 Adobe Post Script	☐		
	2.2.5.2 Adobe PDF	☐		
	2.2.6 Structured formats	☐		
	2.2.6.1 HTML	☐		
	2.2.6.2 SGML	☐		
	2.2.6.3 XML	☐		
	2.2.7 Audio and video formats	☐		
	2.2.7.1 Wave	☐		
	2.2.7.2 Real	☐		
	2.2.7.3 MP3	☐		
	2.2.7.4 AVI	☐		
	2.2.7.5 MPEG	☐		
	Sub-category score		☐	
	2.3 **Version control**		**2.00**	
	2.3.1 Allows past versions of files to be retrieved	☐		
	2.3.2 Changes can be identified	☐		
	2.3.3 Changes can be compared	☐		
	Sub-category score		☐	
	Category Score			☐
3.0	**Metadata**			**12.00**
	3.1 **Real time updating and indexing of accepted content**		**1.50**	
	3.1.1 Features available	☐		
	Sub-category score		☐	

Figure A1.

(continued)

	Checklist Categories		Weight	
3.2	**Metadata schemas supported**		**5.00**	
	3.2.1 Dublin Core	☐		
	3.2.2 EAD	☐		
	3.2.3 MARC 21	☐		
	3.2.4 LOM	☐		
	3.2.5 METS	☐		
	3.2.6 MODS	☐		
	3.2.7 VRA Core Categories	☐		
	3.2.8 MPEG-7	☐		
	Sub-category score			
3.3	**Add/delete customized metadata fields**		**1.50**	
	3.3.1 Feature available	☐		
	Sub-category score			
3.4	**Set default values for metadata**		**0.50**	
	3.4.1 Feature available	☐		
	Sub-category score			
3.5	**Supports Unicode character set for metadata**		**1.50**	
	3.5.1 Feature available	☐		
	Sub-category score			
	Category Score			
4.0	**Search Support**			**14.00**
4.1	**Full text**		**4.00**	
	4.1.1 Boolean logic	☐		
	4.1.2 Truncation/wild cards	☐		
	4.1.3 Phrase	☐		
	4.1.4 Proximity	☐		
	Sub-category score			
4.2	**Search all descriptive metadata**		**2.50**	
	4.2.1 Boolean	☐		
	4.2.2 Truncation/wild cards	☐		
	Sub-category score			
4.3	**Search selected metadata fields**		**1.00**	
	4.3.1 Feature available	☐		
	Sub-category score			
4.4	**Browse**		**2.00**	
	4.4.1 By author	☐		
	4.4.2 By title	☐		
	4.4.3 By issue date	☐		
	4.4.4 By subject term	☐		
	4.4.5 By collection	☐		
	4.4.6 By customized fields	☐		
	4.4.7 Browse by more than 1 field	☐		
	Sub-category score			

(continued)

Figure A1.

	Checklist Categories		Weight
	4.5 **Sort search results**		**0.50**
	4.5.1 By author	☐	
	4.5.2 By title	☐	
	4.5.3 By issue date	☐	
	4.5.4 By relevance	☐	
	4.5.5 By other	☐	
	Sub-category score		
	Category Score		
5.0	**Access Control and Privacy**		**7.00**
	5.1 **Password administration**		**2.00**
	5.1.1 System-assigned passwords	☐	
	5.1.2 User selected passwords	☐	
	5.1.3 Forgotten password retrieval function	☐	
	Sub-category score		
	5.2 **User management**		**1.50**
	5.2.1 Add user profile	☐	
	5.2.2 Edit user profile	☐	
	5.2.3 Delete user profile	☐	
	Sub-category score		
	5.3 **Limit access at different levels**		**1.50**
	5.3.1 File/object level	☐	
	5.3.2 Collection level	☐	
	Sub-category score		
	5.4 **User roles**		**2.50**
	5.4.1 Allows definition of different user groups	☐	
	5.4.2 Limits access by role	☐	
	5.4.3 Allows collection to be customized for each role	☐	
	Sub-category score		
	5.5 **Access management**		**1.50**
	5.5.1 IP source address filtering	☐	
	5.5.2 Proxy filtering	☐	
	5.5.3 Credential-based access	☐	
	Sub-category score		
	5.6 **Security methods**		**1.00**
	5.6.1Encryption	☐	
	5.6.2 Digital signatures	☐	
	Sub-category score		
	Category Score		
6.0	**Report and Inquiry Capabilities**		**3.00**
	6.1 **System generated usage statistics**		**6.00**
	6.1.1 Feature available	☐	
	Sub-category score		

Figure A1.

(continued)

	Checklist Categories		Weight	
	6.2 **Usage reports**		**4.00**	
	6.2.1 Report timeline specification	☐		
	6.2.2 Report fields customization	☐		
	6.2.3 Report templates	☐		
	Sub-category score			
	Category Score			
7.0	**Preservation**			**5.00**
	7.1 **Persistent document identification**		**5.00**	
	7.1.1 System assigned identifiers	☐		
	7.1.2 CNRI Handles	☐		
	Sub-category score			
	7.2 **Quality control**		**3.00**	
	7.2.1 Feature available	☐		
	Sub-category score			
	7.3 **Prescribed digital preservation strategy**		**2.00**	
	7.3.1 Feature available	☐		
	Sub-category score			
	Category Score			
8.0	**Interoperability**			**10.00**
	8.1 **OAI-PMH**		**5.00**	
	8.1.1 Feature available	☐		
	Sub-category score			
	8.2 **Z39.50 protocol compliant**		**3.00**	
	8.2.1 Feature available	☐		
	Sub-category score			
	8.3 **Research protocols**		**2.00**	
	8.3.1 Dienst	☐		
	8.3.2 SDLIP	☐		
	Sub-category score			
	Category Score			
9.0	**User Interface**			**8.00**
	9.1 **Modify interface "look and feel"**		**5.00**	
	9.1.1 Feature available	☐		
	Sub-category score			
	9.2 **Apply a customized header/footer to static or dynamic pages**		**3.00**	
	9.2.1 Feature available	☐		
	Sub-category score			
	9.3 **Supports multi-lingual interfaces**		**2.00**	
	9.3.1 Feature available	☐		
	Sub-category score			
	Category Score			

(continued)

Figure A1.

	Checklist Categories		Weight	
10.0	**Standards Compliance**			**10.00**
	10.1 **Structured document forrmats e.g. XML, SGML**		**3.00**	
	10.1.1 Feature available	☐		
	Sub-category score			
	10.2 **Metadata formats e.g. Dublin Core**		**3.00**	
	10.2.1 Feature available	☐		
	Sub-category score			
	10.3 **Text document formats e.g. Unicode**		**2.00**	
	10.3.1 Feature available	☐		
	Sub-category score			
	10.4 **Image formats e.g. TIFF**		**2.00**	
	10.4.1 Feature available	☐		
	Sub-category score			
	Category Score			
11.0	**Automatic Tools**			**4.00**
	11.1 **Metadata entry system**		**5.00**	
	11.1.1 Feature available	☐		
	Sub-category score			
	11.2 **Generation of**		**5.00**	
	11.2.1 Search indexes	☐		
	11.2.2 HTML pages	☐		
	11.2.3 Reports	☐		
	Sub-category score			
	Category Score			
12.0	**System Support and Maintenance**			**9.00**
	12.1 **Documentation/manuals**		**4.00**	
	12.1.1 Feature available	☐		
	Sub-category score			
	12.2 **Mailing lists/discussion forums**		**2.00**	
	12.2.1 Feature available	☐		
	Sub-category score			
	12.3 **Bug track/feature request system**		**1.00**	
	12.3.1 Feature available	☐		
	Sub-category score			
	12.4 **Help desk support**		**3.00**	
	12.4.1 Feature available	☐		
	Sub-category score			
	Category Score			
	Consolidated Score			

Figure A1.

Note: Category weights sum to 100, while subcategory weights sum to 10

(1) Place a check mark for each item that exists in the software package being evaluated.

(2) Within each subcategory:
   - multiply the number of check marks by the subcategory weight. This weight also represents the maximum score for that subcategory; and
   - divide this resulting number by the number of items in the subcategory to obtain the subcategory score.

(3) Within each category, sum all subcategory scores, divide the total by 10, and multiply the resulting number by the category weight to obtain the category score.

(4) To obtain the consolidated score, sum all category scores.