SURVEY RESULTS

EXECUTIVE SUMMARY

Introduction

Open source software (OSS) "licenses must permit non-exclusive commercial exploitation of the licensed work, must make available the work's source code, and must permit the creation of derivative works from the work itself." [St. Laurent, Andrew M. (2008). Understanding Open Source and Free Software Licensing. O'Reilly Media, p 8. ISBN 9780596553951].

The emergence of OSS increases collaboration among research libraries, providing greater control of library tools, as well as improving usability and quality of library resources. This collaborative approach fits neatly with the knowledge and resource sharing ideology of libraries. While OSS is ostensibly "free," adoption of OSS within an organization is not without significant support, integration, and development costs.

The purpose of this survey was to study ARL member libraries' adoption and/or development of OSS for functions such as an integrated library system (ILS), discovery layer, electronic resource management, inter-library loan, digital asset management, institutional repository, course reserve, streaming media, study room scheduler, digital preservation, publishing, floor maps, data warehouse, and other library-related purposes. We wanted to understand organizational factors that affect decisions to adopt OSS, the cost of OSS, and the awareness of OSS systems already in use. With regard to development of OSS, we wanted to understand: 1) research libraries' policies and practices on open sourcing their code; 2) the frequency of research library contributions to open source projects; 3) the reluctance of research libraries to make their code openly available; and 4) the most common benefits and challenges encountered when research libraries open source their code.

This survey was distributed to the 125 ARL member libraries in February 2014. Seventy-seven libraries (62%) responded to the survey by the March 17, 2014 deadline.

Library IT Staff

The 66 responding academic and public libraries reported between two and 50 staff with IT responsibilities as all or part of their duties, with an average of 16 and a median of 14. Three national libraries reported between 130 and 350 IT staff. This bimodal distribution is stark, with the national libraries an order of magnitude larger than their university counterparts. Despite this difference in staff size, we find no statistically significant differences in the relative participation in OSS projects.

Seventy respondents (91%) develop software in-house. Of those, the most common software development practices include using version control (86%) and performing usability tests (86%). The least common practices include the use of independent quality assurance (24%), adherence to a formal, written code reuse policy (10%), and the presence of a committee or working group to encourage code reuse (7%). The most common other software practices mentioned by respondents were agile/scrum development methodologies (5 of 15 respondents) and pair programming (2 respondents). Most respondents reported that their library IT staff are encouraged to experiment with new technologies (75 or 99%), and prototype potential projects (62 or 82%).

As expected, we found a strong positive correlation between staff size and support for software development best practices (particularly creation of software documentation and specifications, creation of user documentation, performing code reviews, using version control, practicing casual code reuse, and standardizing development by utilizing a common framework).

When asked how users give feedback to IT staff, several findings emerged:

- Library employees most commonly give feedback through a helpdesk or bug tracking system (69 respondents, or 91%) and by emailing or calling the system manger/developer directly (67 or 88%).
- Employees of the parent institution give feedback through a form on the library website (54 or 71%), through subject librarians (44 or 59%), by emailing or calling the system manger/developer directly (39 or 51%), and through a helpdesk or bug tracking system (35 or 46%).
- In-library patrons most commonly give feedback through a form on the library website (59 or 78%) and through subject librarians (58 or 76%).
- Remote users most commonly give feedback through a form on the library website (60 or 79%), and through subject librarian (49 or 64%)

In-library users and remote users most commonly use the same feedback methods, suggesting that proximity to the physical library may not significantly impact feedback channels.

In our review of organizations that contribute to open source projects, software development staff ranged from one or two to as many as 14. While organizations that contribute to large scale, formal open source projects were clearly investing heavily in programming staff, it was also clear that a few organizations who didn't have resources for large technology staffs could still contribute to projects with as few as one programmer. The median number of staff reported as working on OSS projects was two, with an average of nearly four.

Organizational structures varied considerably. Within smaller organizations, single programmers are often located in library systems or web units. Within larger organizations, software development staff are often clustered together in application development units located in digital library, digital projects, or library technology branches of the organization.

Library Software

The survey asked respondents to provide information about the type of software used for various library purposes. All 76 respondents use one or more vended products, 72 identified types of open source software used by the library, and 50 identified software that was built in-house. Below are some of the highlights of the range of applications being used.

- Fifty-eight respondents (76%) use a vended, locally hosted integrated library system (ILS). No respondents use an ILS built in house, but five use an open source ILS.
- Forty-five respondents (59%) use a vended, locally hosted interlibrary loan (ILL) system and 29 (38%) license a software as a service (SaaS) ILL system.
- Forty-nine respondents (64%) use a SaaS discovery layer, 17 (22%) use a vended, locally hosted discovery layer, and 10 (13%) use a discovery layer that is built in house. Several respondents indicated that their discovery layer was both a vended, locally hosted system and also built in house, suggesting significant customizations to a vended product.
- Forty-seven respondents (62%) use a locally hosted and supported OSS institutional repository.
- Forty respondents (53%) use a locally hosted and supported OSS digital preservation system.
- Fifty-one institutions (67%) have adopted a system that is open source and supported by a third party.
- The most commonly built in-house systems were floor maps (28 respondents) and digital asset management systems (19 respondents).

- The most frequently adopted OSS systems include institutional repositories (52 respondents), blogging (50 respondents), digital preservation (47 respondents), and publishing (40 respondents).

OSS Adoption

Seventy-four of the 76 responding libraries (97%) report having adopted open source software. Of these, only five (plus one parent institution) have a formal written policy related to adoption of OSS. Twenty-five libraries (34%) have an informal policy, but the other 43 (59%) have no OSS adoption policy. Several respondents reported that policies were currently being created, but could not be shared at the time of their response.

Most respondents indicated their institution had neither a sustainability strategy (50 of of 71 respondents, or 70%) nor an exit strategy (53, or 75%). Reported strategies include minimizing customizations, providing sufficient staffing with needed expertise, and only adopting systems with good documentation and an active community. More than half of the respondents who commented on their exit strategy emphasized the criticality of data migration (8 of 15 relevant comments).

Survey respondents were then asked to identify the system they had most recently adopted and to provide the number of staff and hours required to implement that system. A wide variety of projects were identified, the most common being Drupal, Blacklight, Omeka, and DSpace. Respondents reported from one to eight staff members dedicated to implementation, with a mean and median of three staff. The number of hours required for initial implementation varied dramatically, ranging from 0.75 hours to 9,000 hours with a mean of 573 hours and a median of 160 hours.

Respondents were asked to identify the open source system they most recently adopted that is still in production and to describe the resources needed to support that system. For most respondents, the system referred to in this question was the same system described in the implementation question above. The number of staff required to maintain this system ranges from 0 (for a digital exhibit) to 10 (for a CMS) with a mean of 2.1 and a median of 2. The number of hours required to support the same system ranged from 0 (for the exhibit) to 512 (for a digital repository) per month, with a mean of 68 hours and a median of 20 hours.

Only 16 of 72 respondents (22%) were able to track the costs of either adopting or contributing to an OSS system. Ten respondents who could track the cost of their most recently adopted OSS system reported that expenses ranged from $400 to over $600,000 and, in some cases, represented a multi-year investment. These funds covered a variety of expenses including staff time, hosting, travel, and consulting. The nearly universal source of funding for adopting or contributing to an OSS system was the library's operating budget (69 of 70 respondents, or 99%). A few had additional funding from grants, their university, or a consortium. One ArchivesSpace project received only consortium and grant funding.

The survey asked respondents to describe three benefits and three challenges associated with adopting OSS. The most common benefit is the ability to customize the software (50 responses). Other common themes include low cost or time to implement (27 responses) and the association with an active community (27 responses). The most common challenge was the need for highly skilled staff who could provide support for the OSS system (40 responses). Other commonly cited challenges include poor documentation (19 respondents), a need for additional training or expertise (16 respondents), and substandard development practices (12 respondents).

OSS Contribution

Fifty-six of the responding libraries (78%) have contributed resources to an open source project. The number of projects contributed to by each library ranges from 1 to 20, with an average of 4.6 and a median of 3. Thirty-two libraries report being the primary code contributor for at least one project; a different set of 32 libraries (with significant overlap) identified themselves as the original developer of an open source project.

Commonly reported examples of projects include DSpace (12 respondents), Fedora (11 respondents), Hydra (9 respondents), Kuali (6 respondents),

Blacklight (5 respondents), and ArchivesSpace (4 respondents). Below are some of the highlights of library contributions to the projects.

- The most common contributions involved code or developer time (47 respondents), funding (36 respondents), hosting (36 respondents), and testing (8 respondents).
- Across all types of contributions, the most common types of projects included institutional repositories (38 respondents), digital preservation (30 respondents), digital asset management (22 respondents), discovery layer (15 respondents), publishing (13 respondents), authentication/identity management (10 respondents), and electronic resource management (10 respondents).
- Code was most commonly contributed to projects on institutional repositories (32 respondents), digital preservation (22 respondents), digital asset management (20 respondents), and discovery layers (11 respondents).
- Digital preservation and institutional repository projects most often received funding via monetary contributions (19 and 18 respondents, respectively), followed by digital asset management projects (8 respondents).
- Hosting was contributed most often to digital preservation projects (9 respondents), followed by repository and publishing projects (5 respondents each).

When asked about reasons for open sourcing their project, respondents listed the following as being "important" or "very important": a belief that open sourcing would lead to better software (30 respondents), a desire to contribute to an open source community (29 respondents), and shared effort in development and quality assurance of the project (27 respondents).

Sixty respondents (78%) develop plugins, extensions, or customizations for a library-related proprietary or vended system. Of these, 31 (54%) indicated vendors allowed them to distribute the code under an open source license.

As was the case with OSS adoption policies, 44 respondents indicated their library has no policy in place for contribution to open source projects, while 20 respondents have an informal policy. Thirty-four respondents stated that they have no tech transfer policy, while 23 respondents indicated that their parent institution has a formal, written tech transfer policy.

Respondents were asked to describe three benefits and three challenges associated with **contributing** to OSS. The benefit most commonly cited was engagement in the open source community (38 respondents). Other common themes included control of product features and direction (25 respondents), and recognition/reputation (14 respondents). The most common challenge was allocating sufficient staff time to make meaningful contributions (24 respondents). Other commonly cited challenges included writing generalized software for use by a larger community, and securing the financial resources needed to support the open source project and community (7 respondents each).

Since open source project members are rarely collocated, a variety of tools have been employed to help coordinate development efforts. Common tools used include shared version control (37 of 45 respondents, or 82%), an issue tracker (36 or 80%), a mailing list, (32 or 71%), and a wiki (25 or 56%). Forty-one respondents (79%) use a public repository or forge to share their open source code; Github was by far the most common (38 of 41 respondents, or 93%).

The most common licenses used by respondents were GNU Public License v3 (16 respondents), Apache, and Creative Commons (15 responses each).

Respondents were asked to rank a set of success indicators in terms of their importance for the respondent's institution. A significant number (41 or 80%) identified as most important that the functionality better suits their institution's needs.

Respondents were asked if any of their in-house software could have been, but has not yet been, released under an open source license. The 53 respondents (69%) who answered in the affirmative expressed concerns about the staff time commitment required to support the community (41 or 77%), the readiness of code quality for public adoption (39 or 74%), and dependence on other internal systems (30 or 57%).

Conclusion

This survey reveals that nearly all of the responding ARL member libraries are developing custom software and/or adopting one or more open source systems. Contribution to OSS projects is also common, with more than three quarters of respondents actively contributing to OSS projects.

Many respondents expressed a desire on the part of their developers to share with and participate in one or more OSS communities. Larger organizations committed more resources to OSS projects than smaller organizations, but we found no significant correlations suggesting a disproportionate level of commitment to OSS projects as a function of IT staff size. The nearly universal adoption of OSS systems and the high level of contribution to OSS projects may suggest that adoption of and contribution to OSS projects has entered the mainstream for libraries. Simply stated, libraries that develop software also predominantly contribute to OSS projects.

The results of this survey suggest that libraries view organizational behaviors surrounding the adoption of open source software separate from contribution to OSS projects. For example, while respondents view OSS adoption as a means of saving time and resources, contributing to OSS projects is viewed as being advantageous for different reasons, primarily engagement in an OSS community. For developers, the sense of social involvement in a community represented by an OSS project can be a positive source of professional satisfaction, ultimately leading to greater productivity and a return on investment for the library.

Control of software emerged as a theme common to both adopting and contributing to OSS projects. Those adopting OSS systems felt that access to source code gave them greater control, allowing them to change the software as needed, rather than being subject to the whims of a proprietary solution. Those that contributed to OSS projects felt that they gained greater opportunity to influence product direction, especially with respect to product features. In both cases, they perceived a sufficient benefit to their overall productivity to justify the expense of their involvement (as adopters, contributors, or both) in OSS systems.

Page intentional left blank.

SURVEY QUESTIONS AND RESPONSES

The SPEC Survey on Open Source Software was designed by **J. Curtis Thacker**, Discovery Systems Manager at Brigham Young University's Harold B. Lee Library, **Dr. Charles D. Knutson**, Associate Professor of Computer Science at Brigham Young University, and **Mark Dehmlow**, Program Director for Information Technology at the University of Notre Dame's Hesburgh Libraries. These results are based on data submitted by 77 of the 125 ARL member libraries (62%) by the deadline of March 18, 2014. The survey's introductory text and questions are reproduced below, followed by the response data and selected comments from the respondents.

Open source software (OSS) is software that adheres to the following principles: "open source licenses must permit non-exclusive commercial exploitation of the licensed work, must make available the work's source code, and must permit the creation of derivative works from the work itself." [St. Laurent, Andrew M. (2008). *Understanding Open Source and Free Software Licensing*. O'Reilly Media, p 8. ISBN 9780596553951].

The emergence of OSS has increased collaboration among research libraries, providing greater control of library tools, as well as improving usability and quality of library resources. This collaborative approach fits neatly with the knowledge and resource sharing ideology of libraries. While OSS is ostensibly "free," adoption of OSS within an organization is not without significant support, integration, and development costs.

The purpose of this survey is to study ARL member libraries' adoption and/or development of OSS for functions such as ILS, discovery layer, electronic resource management, inter-library loan, digital asset management, institutional repository, course reserve, streaming media, study room scheduler, digital preservation, publishing, floor maps, data warehouse, or other library-related purposes. We would like to understand organizational factors that affect decisions to adopt OSS, the cost of OSS, and the awareness of OSS systems already in use. With regard to development of OSS, we would like to understand: 1) research libraries' policies and practices on open sourcing their code; 2) the frequency with which research libraries contribute to open source projects; 3) whether research libraries are reluctant to make their code openly available; and 4) the most common benefits and challenges encountered when research libraries open source their code.

IN-HOUSE SOFTWARE DEVELOPMENT

1. How many individuals in your library are responsible for information technology as all or part of their duties? ("Library IT staff" could be a well-defined department or a small part of one person's duties.) N=69

All Respondents

Minimum	Maximum	Mean	Median	Std Dev
2	350	31.78	15.0	72.43

Academic Library Respondents N=65

Minimum	Maximum	Mean	Median	Std Dev
2	50	15.89	14.0	10.62

Number of IT Staff	Responses
2	1
3	1
4	2
5	3
6	3
7	3
8	4
9	5
10	3
11	1
12	3
13	2
14	3
15	3
16	3
17	2
18	4
19	4
20	2
21	1
23	1
25	3
26	1
30	1
36	1

Number of IT Staff	Responses
38	2
40	1
50	2

Nonacademic Library Respondents N=4

Minimum	Maximum	Mean	Median	Std Dev
30	350	190.00	190.00	139.52

Number of IT Staff	Responses
30	1
130	1
250	1
350	1

2. Do library IT staff develop any in-house software? N=77

Yes	70	91%
No	7	9%

If yes, which of the following software development practices do library IT staff employ? Check all that apply. N=70

Usability testing	60	86%
Version control	60	86%
Software documentation and specifications	55	79%
Iterative releases (i.e., small and frequent releases)	53	76%
Reuse of in-house code libraries	52	74%
Reuse of shared framework(s)	51	73%
Casual code reuse between developers	50	71%
User documentation	49	70%
Developer unit testing	44	63%
Accessibility testing	39	56%
Code reviews	38	54%
Coding style guidelines	35	50%
Code commenting guidelines	33	47%
Independent quality assurance	17	24%
Reuse of purchased code libraries	13	19%
A formal written code reuse policy	7	10%
A committee or working group to encourage reuse and oversee shared code	5	7%
Other software development practice(s)	15	21%

Please briefly describe the other software development practice(s) your library IT staff employ. N=15

Acceptance testing, pair programming, community code review, continuous integration, DevOps practices

Agile/Scrum project management practices

Agile development

Agile development methodology with active involvement of customer

Agile Project management

Agile Scrum development methodology. Also note that not all practices checked above are applied universally across all projects.

Continuous integration, bug/enhancement tracking, backlog management

Deployment strategies, such as Capistrano

Experimental software as part of research projects

Functional testing. Virtualized development environments and code driven environment configuration. Design patterns. Agile approach, trying to implement a 2–3 week cycle for milestones. Frequent standups, not daily but certainly when issues arise. Iterative development with incremental feedback.

Informal usability test

Modify open source code for library use.

Pair programming

Pair programming, interaction design (personas, user stories, prototyping), TDD

Security checks, penetration testing

3. Which of the following activities are library IT staff encouraged to participate in? Check all that apply. N=76

Experimenting with new technologies	75	99%
Prototyping for potential projects	62	82%
Rewriting existing systems to make them easier to support	57	75%
Collaborating on projects that are not part of their specific responsibility	56	74%
Other related activity	10	13%

Please briefly describe the other related activity. N=10

Collaborating with developers outside the Libraries, participating in open-source developer communities, attending developer users' groups meetups.

Configuring, customizing, and extending existing systems.

DevOps work to support operations staff.

Existing systems are rewritten only when there is a need.

Inter-campus work, marketing department and ITS

Other responsibilities as assigned/needed.

Professional conferences

Streamline services, decommission paid services, security review.

Training on related emerging software technologies and platforms.

We work to keep applications supportable in the library by choosing technologies and languages that can be supported by more than one person in IT, and through cross training on those technologies.

4. How do users of library systems give feedback to your library IT staff? Check all that apply. N=76

Feedback Method	Library employees	Institution employees	In-library patrons	Remote users	N
Through a helpdesk or bug tracking system	69	35	25	31	71
Emailing or calling the system manager/developer directly	67	39	16	23	68
Through a web form built into the library website	48	54	59	60	65
Through subject librarians	33	44	58	49	65
There is no established method	—	—	1	—	1
Other method	5	3	6	6	8
Number of Responses	76	69	75	71	76

If you selected "Other method" above, please specify the user group and briefly describe that method. N=12

"Contact us" link and Chat

Emails or chat notes or phone messages forwarded by other library employees.

In person

In person discussions [with library employees]

Our public feedback takes place through email to support web sites, or notes in suggestion boxes. Our system user feedback takes place through the Help Desk.

Service teams for our major brands who help assess requests for features, problems, projects, etc.

Through library public service staff (not all of them necessarily subject librarians).

User research, informal conversations with members of various groups

We have a User Experience department that employs several methods for gathering feedback of existing services, as well as feedback and input on services as they are being implemented.

We have an extensive release testing process that involves faculty and staff throughout the libraries.

We no longer have a web form for tech support; it was replaced with a web helpdesk ticketing system. The IT ticketing

system has many different categories of help, and it is used by a variety of campus departments. Help requests are triaged to the appropriate campus department based on need.

We occasionally hold focus group sessions with student users (generally undergraduates). These are sometimes very informal introductions to prototypes on which we gather first-reaction comments to inform further development, at other times, these are more structured formal feedback opportunities.

SYSTEMS BUILT IN-HOUSE THAT AREN'T OPEN SOURCED

5. Has your library built in-house any library-specific systems that could be, but have not been, released as open source? N=77

Yes	53	69%
No	24	31%

If yes, what are the primary reasons for not releasing it as open source? Check all that apply. N=53

Concerns about staff time commitment required to support the community	41	77%
Concerns that the code quality is not ready for public adoption	39	74%
Dependence on other internal systems	30	57%
It didn't occur to us	7	13%
Seeking to license or sell the system	2	4%
A competitive desire to have the best system	1	2%
Other reason(s)	12	23%

Please briefly describe the other reason(s) for not open sourcing the system. N=12

Highly customized to address local requirements.

Lack of clarity about campus policies for licensing and intellectual property ownership.

Legal considerations.

Narrow niche applications where a community is unlikely to develop.

Not approved for release.

Not documented for external audiences.

Often these systems reflect local practices. We've not viewed them as useful beyond our local environment.

Planning to release a service as open source, working on appropriate licensing language at this time.

Security

Security concerns related to embedded information.

Technology Commercialization Office needs to review any software developed at the university.

Time needed for review of and compliance with licenses of third-party components.

CUSTOMIZING PROPRIETARY SYSTEMS

6. Does your library develop plugins, extensions, or customizations for any proprietary or vended systems? N=77

Yes	60	78%
No	17	22%

If yes, do those vendors allow the code you developed to be openly distributed with OSS licensing? N=57

Yes	31	54%
No	26	46%

Comments N=16

Customizations are specific to our institution's unique requirements and would not be generally useful to others. Some customizations would not be supported by organization for security and support reasons.

Ex Libris allows/encourages development and customization of their systems, but sharing is limited to other Ex Libris user institutions via CodeShare on the password-protected Ex Libris EL Commons web site.

In some cases, we are not sure, because we have not specifically asked the vendor. In the case of our ILS vendor, their willingness to have our code openly distributed depends upon how much proprietary information about the system would be divulged by the new software, i.e., the nature of the software and how it interacts with the proprietary system.

Most do allow for this. Or, they at least have an established community of their customers where code can be shared. We attempt to write code that is mostly generalizable to any like system, in order to allow ourselves the flexibility to changes systems later on with fewer dependencies on custom development.

Not all our vendors allow this. Some applications would reveal proprietary information about the data model used in vendor product.

Not sure if it's allowed (haven't asked).

Some allow this, some do not.

Some vendors allow it, others do not. Ability to redistribute is not a major factor in determining whether we develop plugins, extensions, or customizations.

Some vendors do, some vendors don't.

The library has developed plugins for use with its proprietary ILS software (Voyager) and has shared the plugins with other libraries. They are considered a federal employee product, therefore public domain.

The library IT staff has plans to develop plugins, extensions, or customization for the ILS. The ILS vendor does allow APIs to be openly distributed.

Unsure [whether vendor allows this].

We do provide the extensions without a license but we include a disclaimer.

We have a couple of vendors that have taken contributions from our teams but that code is not openly distributed with OSS licensing.

We primarily build them for us and share them if we can. Some vendors allow for semi-open sharing.

With the signing of appropriate releases and/or agreements.

LIBRARY SOFTWARE

7. Please identify the type of software used by your library for each of the following purposes. Check all that apply. N=76

Purpose	OSS (locally hosted, locally supported)	OSS (locally hosted, supported by a third party)	OSS (hosted and supported by a third party)	Vended product (locally hosted)	Vended product (hosted by the vendor or SaaS)	Built in-house	N/A	N
Inter-library loan	2	—	1	45	29	4	3	76
Institutional repository	47	1	6	5	12	14	7	76
Digital preservation	40	10	7	11	3	15	19	76
ILS	3	1	2	58	17	—	1	75
Discovery layer	16	2	3	17	49	10	2	75
Course reserve	2	—	2	43	16	12	7	75
Electronic resource management	8	—	1	18	38	13	3	74
Streaming media	16	1	—	33	18	5	12	74
Blogging	38	2	13	11	8	1	9	74
Authentication/identity management	25	7	8	33	8	11	7	74
Digital asset management	33	3	2	20	11	19	9	73
Study room scheduler	17	—	1	13	20	13	14	73
Publishing	36	3	4	4	10	5	19	73
Link resolver	5	1	4	22	43	7	3	73
Floor maps	8	—	—	8	5	28	28	71
Web analytics	15	2	7	10	47	4	—	71
Data warehouse	11	1	2	7	4	10	43	69
ELMS	4	1	2	11	6	2	45	68
Data analysis	6	1	1	17	11	8	36	68
Visualization	10	—	—	15	8	3	40	67
Other purpose	13	1	—	4	3	9	10	31
Number of Responses	70	22	40	76	73	50	67	76

If you indicated above that the library is using any software for an "Other purpose," please briefly describe that purpose. N=25

Archival description software (ICA-AtoM for archival finding aids)

Archival Management for managing archival data.

Citation Fox and IL Fox

Content management system

Course reserve is Blackboard, hosted by university IT, not the library.

Database software (MySQL), Web Server (Apache), Exhibits (Omeka), Timeline & Map web support (Neatline)

Electronic Finding Aids: currently use Archon, will move to ArchivesSpace in the future.

Enterprise service bus and rapid application development environment afforded by Kuali Rice.

FYI, we are considering vended product/hosted by vendor to include the university's central IT unit (Office of the Chief Information Officer) and central academic computing unit (Office of Distance Education and E-Learning).

Here are some top software products the Libraries have developed to fulfill our needs: research consultation services, equipment management, trouble ticket, feedback, hours, event administration, news/alerts, reference transactions, spam blocking, reminders. Also, we have a vendor product for single-sign on for our ILS. Lastly, there are, additionally, more campus central IT run services that the Libraries use.

Just wanted to note an additional dimension to consider. We make use both of very library-specific software primarily managed by the Libraries but are also heavy users of software provided by our university's central IT dept. In some cases the relationship is somewhere in between a locally hosted and vendor hosted situation.

Many of the choices above do not allow for accurate categorization of our environment.

Monitoring, performance analysis, metrics, digital signage

Note: Dataverse (Data Warehouse) and geospatial software (Data Analysis) on shared consortial system: From Scholars Portal, of the Ontario Council of University Libraries.

Offsite storage inventory, RFID, self-checkout.

Omeka for online exhibits

Other purpose is Digital Collections application and ContentDM for metadata management.

Persistent identifier software

Research guides/FAQs, digital exhibits, EAD repository, staff directory, Database A-Z

Resource annotation and analysis tool (RUanalytic). Metadata and resource handling application (OpenWMS) and ETD submission system (RUetd)

Scientific data analysis, text mining

Social media archiving, and social media display/sharing

Subject-specific databases/portals, electronic access

We also have several productivity tools that are small productivity applications, such as tools for replacement materials

workflows, another for reformatting. Our subject pages are driven by the MyLibrary toolkit. We use Library a la Carte for subject guides.

We use OSS and in-house software for many other needs: lots of back end server stuff like sharing data between systems, and frontend custom displays for various resources.

8. Please indicate how important each of the following software selection criteria is to your library. Please make one selection per row. N=76

Criteria	1 Not Important	2	3	4	5 Very Important	N
Functionality that best meets our needs	—	—	1	14	61	76
Staff time to support	—	2	13	35	26	76
Control and customizability	—	1	13	36	26	76
Monetary cost for support and maintenance	—	—	14	40	22	76
Staff time to implement	—	3	21	31	21	76
Monetary cost for implementation and licensing	—	2	14	31	27	74
Other criteria	1	—	2	6	12	21
Number of Responses	1	6	42	65	70	76

If you indicated above that the library is using any "Other criteria" to select library software, please briefly describe the criteria. N=17

Academically developed and controlled to reduce risk. We do buy vendor solutions but with intention and critical analysis due to the amount of data we have and priority to preserve and make that information available.

ADA compliant, standards based, interoperable with other systems, meets security standards

Adoption of the software in the wider (library) community. Whether or not the software is actively being maintained.

Compatibility with existing systems

Compliance with industry standards for system interoperation

Integration with complex information environment; ability to extend software beyond library to provide services to other departments and institutions; opportunities afforded for professional development in open- and community-sourced software.

Integration with existing systems

Integration with other library systems. Community of software users and evidence of development.

Interoperability with existing systems. Community around an OSS project.

Interoperability with other systems; sustainability

Is it open source?

It is important for any systems to meet accessibility standards.

Safety and security of the software (impact on IT security at the library)

Software quality and reliability

Use of open data standards

Vendor responsiveness for vended products or a robust user community or user groups for OSS.

We try to insure that all components of our cyberinfrastucture, whether developed in house or not, work well together to fit within the RUcore architectural framework. All tools and services can then be managed together and receive upgrades/enhancements on the same schedule. Our commercial ILS, Sirsi/Dynix does not support this and one IMPORTANT reason we are moving to Kuali OLE is the ability to integrate all our cyberinfrastructure into a coherent platform where the focus can be an integrated approach to user needs.

Please select the correct statement about the use of OSS at your library. N=76

Our library is using open source software	74	97%
Our library is NOT using any open source software	2	3%

If your library is using open source software, you will continue to questions about OSS Policies.

If your library is NOT using any open source software, you will skip to the screen Library Doesn't Use OSS.

OSS POLICIES

9. Please indicate the kinds of policies your institution has related to OSS. Check all that apply. N=73

OSS Policy Content	Formal, written library policy	Formal, written parent institution policy	Informal library policy	Informal parent institution policy	No policy	N
Adoption of OSS developed elsewhere	5	1	25	7	43	73
Development of OSS in-house	3	4	20	10	44	73
Contributing resources to OSS projects	4	5	20	6	44	73
Technology transfer	2	23	4	8	34	69
Number of responses	7	24	32	16	59	73

Comments N=8

I am not aware of any official or documented policy regarding OSS at the institution at this time.

The library has policies and procedures for making library-produced open source code available outside the library. The policies are currently under editorial revision and are expected to be released later in 2014.

Not aware of university policy though it may exist.

Our library informally supports and greatly encourages IT staff to use and contribute to OSS projects.

We are just beginning to develop policies in this area.

We have no formal policies with regards to OSS. We are pragmatic in our approach to open source software, and compare with vended solutions based on criteria noted earlier in this survey.

We know from experience there is a process, but could not locate the policies.

Whether a commercial vendor or OSS product best meets a given need is determined on a case-by-base basis.

10. Does your institution have either a sustainability or exit strategy related to OSS projects? N=71

Strategy	Yes	No
Sustainability strategy	21	50
Exit strategy	18	53

If there is either a sustainability or an exit strategy, and a document that describes the strategy, please include the document in the Call for Documents at the end of the survey.

If there is a strategy, but no document, please briefly describe the strategy below.

Sustainability Strategy N=15

Informal. Must be sustainable. Implementing department is accountable.

Minimize customization.

Platform review on a regular basis (~five year cycle).

Provide staff support for ongoing development of our open source content management system (Drupal) and ongoing support and development of our institutional repository (if we stay with an open source product after our pilot project).

Staff to support; minimum customization; data management a requirement.

Stated in strategic plan and through staffing, but no formal document.

Supported as a strategic application, that is, assigned as primary responsibility for a group or person in IT.

The closest is the Hydra partner agreement https://wiki.duraspace.org/display/hydra/Hydra+Community+Framework

The Kuali OLE project, not yet in production, is developing a sustainability plan to grow and sustain the software for at least a decade. This includes ongoing support, in cash and in-kind, from partners, attracting new partners, and partnering with commercial affiliates for software support, training, implementation, and development contributions.

The way in which we contribute and leverage OSS assures that the university has access to all OSS and can continue to maintain, develop, or discard that technology according to our needs and priorities. We are involved in the strategic steering, operational and development of the majority of OSS that we use.

We adopt only OSS projects that have a healthy, active community for collaboration/support. We also choose projects with methods for contributing code back, and with good documentation so in-house work can begin quickly.

We avoid making extreme customizations that are super specific or require extensive changes to the base code, hence sustaining our OSS from one version to another is relatively flexible.

We plan out sustainability in the same manner as other software implementations and development activities.

We will adopt an enterprise OSS system or component only if it is developed within the narrow range of technologies—languages and deployment platforms—in which we have expertise and experience, and only if the system or component is supported by an established, stable community. We follow best practices, particularly around testing and engineering for stability and scalability, in order to minimize support and maintenance costs. We move support out of the development group and into a support group (with partial success).

When adopting OSS or engaging in development of OSS, we look for and/or try to establish a broadly-based community of support in order to mitigate risks of being too dependent on one institution's/individual's resource commitment.

Exit Strategy N=15

Data migration is mandatory.

Exit strategy only concerning ability to export all data and relationships from software.

For the eXtensible Catalog (XC), our exit strategy (which we are now implementing) involves moving all infrastructure support for the software to a library consortium (CARLI) that has been a major partner in developing the system. Our strategy also has included a detailed communication plan for notifying all stakeholders. We have not deployed XC locally. For IR+, we are now discussing possible options for future actions that may include a formal exit strategy.

Informal. Must have a reasonable exit strategy. Implementing department is accountable.

Native export tools/XML, etc. unique to each application

No formal exit strategy. We do choose software with open data standards so that our information can be exported on a whim and used in different software.

Not only with OSS, but with all software systems, we develop such that dependencies are not vendor or product specific, but could allow for replacement of a part of our infrastructure with a like service without having to redesign the whole.

Our data adheres to open standard policies, so if we ever need to migrate out or exit out of the OSS, our data would be compatible with any other system.

The plan will include an exit strategy to allow either end-of-life of the software, or mechanism for turning over software to other interested parties.

To ensure that our data are portable, we require that an open source software be capable of exporting our data in a standard data exchange format.

Use of a software system whether OSS or vended requires data export capability.

We always look at an exit strategy when making a decision about a particular technology solution, regardless of whether it is open source or not.

We keep data and presentation layers separate, so that migration out is easier. We choose OSS with data storage techniques that allow for complete export of all relevant data in a format for easy migration.

We may resort to a hosted/vended product for our institutional repository if we're not satisfied with the results of our pilot project using an open source software repository product.

We regularly evaluate our needs against the technologies we are using and are aware of alternatives. Because we are involved in the strategy and development of most of the OSS, we are also aware of the threats for the OSS that we use. Use of OSS affords us greater time to plan migration or alternative strategies. We have experience and expertise with vended solutions that offered minimum time and therefore forced quick migration and alternative solutions that in some cases have proven to not meet our needs.

REASONS FOR ADOPTING OSS

11. Please identify the open source software that your library has adopted. N=66

Apache, Eventum, Movable Type

Archivists' Toolkit

AutoDewey: software was created at Northwestern University Libraries, adapted at LC.

AWStats, DSpace, Islandora, Fedora Commons, ICA-AtoM, Archivematica, Drupal, Apache Solr, Apache Lucene, Apache, Squid, KeePass, Nagios, PuTTY, MongoDB

Blacklight content management system, Google Map viewer API, California Digital Library Micro Services, Archivists' Toolkit, ArchivesSpace, DSpace, LibStats, Drupal, Omeka, Linux, Apache, LOCKSS

Blacklight, Fedora Commons, DSpace, Handles, WordPress

Blacklight, Fedora Commons

Blacklight, Hydra, Solr, Fedora Commons, DSpace, Opencast Matterhorn, Avalon Media System, Variations Digital Music Library. Many utilities/tools such as ffmpeg, JHOVE, etc.

Digital Library Extension Service (DLXS), Fedora Commons, Omeka, Guide on the Side, Apache, Tomcat, Wikimedia, Linux

Drupal

Drupal, PHP, phpScheduleIt, Blacklight

Drupal, CORAL, Guide on the Side, ArchivesSpace

DSpace (2 responses)

DSpace, Open Journal System (OJS)

DSpace, Drupal

DSpace and several others

DSpace, Fedora Commons, Hippo CMS, Drupal, Open Journal System (OJS)

DSpace, Fedora Commons, Hydra, Apache, MySQL, Solr, Linux, Open Journal System (OJS), Python, R, Ruby, Archivists' Toolkit, ArchivesSpace, WordPress, Drupal, Tomcat

DSpace, Islandora, Fedora Commons, Drupal, Tesseract, ICA-AtoM, Open Journal System (OJS), Open Book System, Manitobia, LOCKSS, PostgreSQL, MySQL, Apache suite of applications, Python, Redmine (Ruby), Git

DSpace, Omeka, MDID

DSpace, Omlaut, WordPress

DSpace, Open Journal System (OJS), and VuFind

DSpace, Open Journal System (OJS), Archivematica, ICA-AtoM, LOCKSS, WordPress, MediaWiki

DSpace, Open Journals System (OJS), eXtensible Text Framework (XTF), Omeka, WordPress, Drupal

DSpace, Fedora Commons, Archivematica, ResourceSpace; Public Knowledge Project (PKP) including Open Monograph Press (OMP), Open Journal System (OJS), Open Conference System (OCS), General Transit Feed Specifications (GTFS), RefStat, Suma, Xibo, Mondo Grinder, phpScheduleIt, software for hours and locations

DSpace. File Analyzer, Archivists' Toolkit, LOCKSS

Fedora Commons

Fedora Commons, Hydra, CORAL, Apache, Puppet

Fedora Commons, Blacklight, Hydra, Solr, Avalon, WordPress, ArchivesSpace (soon), Piwik, MySQL, Apache, Neatline, and many other components for transforming or disseminating information.

Fedora Commons, DSpace, Open Journal System (OJS), Open Conference System (OCS)

Fedora Commons, DSpace, Umlaut, Shibboleth, Xerxes, Blacklight, Vireo, Hydra, Solr. As well we have adopted several OSS, such as Tomcat and Apache, that do not seem to be the focal point of this survey.

Apache, Drupal, Webinator, Fedora Commons, WordPress, Omeka, BuddyPress, Avalon Media System, eXtensible Text Framework (XTF), Bugzilla, Handles, PostgreSQL, PHP, Perl, Linux

Hydra, Blacklight, Solr, Drupal

Hydra, DSpace, Drupal, WordPress, LC Newspaper Viewer, Archivist ToolKit, VireoCat, various open source utilities

Hydra, Fedora Commons, Solr, Blacklight, phpScheduleIt, Open Harvester, WordPress, others

Hydra, Omeka, Drupal, Shibboleth

Islandora

Koha, Fedora Commons, Xerxes, Library a la Carte, WordPress, MyLibrary, eReserves, Blacklight, VuFind, Hydra, CORAL

Linux, django, Python, Solr, Lucene, Nginx, PostgreSQL, various support libraries and toolkits

LOCKSS, Public Knowledge Project (PKP), Omeka, Plone

Lots: Drupal, EZProxy when it was OSS, our web stack, our Moodle LMS, our IR, others.

Open Journal System (OJS) and Omeka; CORAL ERMS

Open Journal System (OJS), DSpace, Omeka

Open Journal System (OJS), Open Monograph Press (OMP), Drupal, WordPress, Dokuwiki, MediaWiki, Islandora, Fedora Commons, Spiceworks, Piwik, Omeka, Archivists Toolkit

Omeka, Avalon, WordPress, Silverstripe, DSpace, Open Journal System (OJS), Open Conference System (OCS)

Open Journal System (OJS)

Open Journal System (OJS), eXtensible Text Framework (XTF), AWStats, Daily Stats, WordPress, Webilizer, GoogleAnalytics, MySQL, PHP

Open Journal System (OJS)

phpScheduleIt, Omeka, WordPress, Archon, ArchivesSpace, Blackligh, SubjectsPlus, Variations, Avalon, Fixity, Assana, MarcEdit, DMPTool, Lucene/Solr, EzProxy, E-Prints

PHP, Blacklight, MongoDB, PostgreSQL, MySQL, Northwestern U Book Viewer, Solr, Lucene, GSearch, Djatoka, Fedora Commons. SciDB, Openstack, django, Openshift, Drupal, CentOS, Cassandra, sqe, Ruby, Python (and libraries), Perl and libraries, many Apache tools, GNU tools, Nagios Open Monitoring Distribution (OMD), Spacewalk, OCS Inventory

PHP, MySQL, Linux, Apache, Drupal

Public Knowledge Project (PKP), Research Project Calculator (Assignment Calculator), ArchivesSpace, Apache, Linux, MySQL, PostgreSQL, Hydra, Blacklight, Fedora Commons, Solr, PersistantURLs (PURLZ), Omeka, Open Journal System (OJS)

Streetprint, DSpace, OS Ticket, Dokuwiki, Guide on the Side

SuraSpace products, SugarCRM, ArchivesSpace

The main library-specific OSS we use: VuFind (and Solr), DSpace, LOCKSS. We make heavy use of other general open source software including Ubuntu, Apache, Tomcat, WordPress, etc.

This list could go on for pages: Apache, Fedora Commons, DSpace, Islandora, WordPress, Drupal, MySQL, Linux, Docker, Redmine, OpenLDAP, VuFind, Arduino IDE, Open Journal System (OJS), Raspbian, OpenOffice, GIMP, etc. We have both servers and desktops running various Linux flavours; nearly every piece of software on them is by nature OSS.

Too many to mention. But here are some: Ubuntu, Apache, PostgreSQL, Python, django, Perl, PHP, Java (openjdk), Solr, jQuery, D3, postfix, Nagios, phpScheduleIt, DSpace, Drupal, MySQL, osTickets.

UCLA MWF, DSpace, MySQL, Apache, PHP, SAMBA, Open SSL, Open SSH, Linux (CentOS and Ubuntu), Sendmail, Solr,, Nutch, Tomcat, WINE, VirtualBox, KeePass, PuTTY, Pidgin, Stat Transfer, WinSCP, 7zip, Firefox, Thunderbird, SPSS, Audacity, MarcEdit, FreeMind, Gimp

Umlaut, Blacklight, Xerxes, Fedora Commons, Solr, DSpace, Drupal, WordPress, Rails, Jenkins, djatoka, OpenLayers, Git, Linux, PHP, Java, Apache, Tomcat, GNU Compiler Collection (GCC)

VuFind

VuFind to develop our discovery layer, Shibboleth for identity management (this is the standard at our parent institution and it has been integrated with library systems).

VuFind, Drupal, CORAL, ARC, Omeka, Solr

VuFind, DSpace, Open Journal System (OJS), Papyrus, Islandora

Webcalendar, Hydra

WordPress, XTF, Omeka, Nagios, PKP OAI Harvester

12. Please indicate how important each of the following reasons for adopting OSS over a competing vended product is to your library. Please make one selection per row. N=72

Reasons	1 Not Important	2	3	4	5 Very Important	N
The functionality of the open source system best meets our needs	—	1	3	14	54	72
Greater control and customizability	1	—	5	26	40	72
Lower monetary cost for implementation and licensing	2	6	25	18	21	72
Lower monetary cost for support and maintenance	2	8	23	25	14	72
Library or institutional policies encourage the use of OSS	27	15	18	11	—	71
Desire to contribute to the library OSS community	6	15	22	18	9	70
Less staff time to implement	2	18	32	10	7	69
Less staff time to support	4	11	31	17	4	67
Other reason(s)	3	—	4	—	3	10
Number of Responses	31	37	65	61	67	72

If you indicated above that the library has other reason(s) for adopting OSS over a competing vended product, please briefly describe the reason(s). N=7

3 Important

Limited availability of software

Ongoing economic sustainability is critical for determination to adopt OSS or a vended product. All public facing web applications must be made accessible for disabled users, so control of this is vital for our institution.

OSS implementations relate to gaps in the vended market.

Staff familiarity with OSS systems.

5 Very Important

Better integration with RUcore cyberinfrastructure.

Freedom to study, copy, modify, and redistribute. Availability of potential staff candidates familiar with free software options. Trust in the respective developer communities.

Resourcing: Leveraging pooled resources within community, which decreases cost for cross training and ensures forward movement and support during staff shortages. Training & retention: staff have a ready network of peers and training opportunities which greatly supports skill building, impact of work, visibility of their work and professional networking.

Additional Comments N=4

For above statements, don't necessarily agree, e.g., "less staff time to implement"—generally takes more time to implement an OSS—so not important is what was selected.

Security, analytics, integration with older systems

We disagree with the statements above that OSS takes less time to implement and less staff time to support, and so were unsure how to respond to them. Saying that they are "not important" to us would be misleading, so we left them blank.

We like our OSS to have a robust developer community.

13. Please identify your most recently adopted OSS system **that has been deployed**, and indicate how many staff and how many hours of staff time were required to complete the initial production deployment. An estimate of the number of hours is acceptable. N=63

OSS System	Staff	Staff hours	Comments
Archivematica			
ArchivesSpace			
ArchivesSpace	2	160	
Archivists' Toolkit	1	100	Customization was contracted out.
Blacklight	3	1500	
Blacklight	4	100	The work was done in two 2-week sprints of about 25 hrs/wk. Part of the experience was getting used to Blacklight as a development environment, in addition to developing the intended discovery piece.
Blacklight	8	9,000 (very rough estimate)	Work on this project spanned many groups and involved work across several units of our organization. This estimate is likely to be fairly inaccurate.
Blacklight			We cannot share cost related information at this time.
Blacklight, Fedora Commons, djatoka, Lucene, Book Viewer	2	Approximately 2,000 hours	OSS allowed team to select best components for specific parts of project to meet project goals of this major development effort. OSS allowed us to greatly customize presentation and functionality. Functional changes are more easily achieved with OSS than a vended product, but of course requires in-house development staff.
CORAL (e-resource management)	1	30	Does not include hours spent with data management from Technical Services; just the time the developer spent.
DAMS – Islandora, Fedora Commons	1	630	
Dokuwiki	1	8	
Drupal	2	500	Change platform for library website.
Drupal	3		
Drupal	3	1000	
Drupal	3	at least 240 hours	Three staff members were involved in the implementation of Drupal, but only a portion of their time for a period of about three months.
Drupal	5	3500	Library website development and deployment.
Drupal	3		Number of hours was not tracked.
DSpace	2	40	

OSS System	Staff	Staff hours	Comments
DSpace	4	200	
DSpace	5	1000	Hours calculated on 4 hours of work per week spread across 5 staff for one year. This relates to a grant project has been going on for several years. 1000 hours is probably a conservative estimate. We have not been formally tracking personnel time for OSS projects.
DSpace	4	200	Mostly one IT staff implementing configurations and changes and two librarian/admin staff making design decisions and testing. Sysadmin time during startup.
Fedora Commons	3	80	
Fedora Commons	4	unknown	
File Analyzer	1	5	
Guide on the Side	3	500	This is a piece of software that we actually developed, so the number of staff hours is very high due to the development time.
Guide on the Side	3	2	Staff included 1 technical resource and 2 librarians.
Hippo CMS	5	2500	Very rough estimate; also includes building the html/CSS for new website from scratch.
ICA-AtoM	3	700	
Islandora	2	many	We can't calculate staff hours with any accuracy, as we haven't been systematically keeping track.
Islandora	2	16	We are counting server build only. Software install was completed by support vendor. We are not counting system evaluation prior to purchase of vendor support or customizations/configuration/initial material ingest.
Islandora	3		Difficult to estimate; deployment bleeds into other issues, such as metadata import, etc.
Islanora	4	160	We have four full time staff developing on the Islandora stack. This includes efforts for Drupal, Solr, and Fedora Commons, which comprise Islandora.
Koha	7	130	For Jerusalem site
LC Newspaper Viewer	4	100	
Linux/Apache/django stack for library widget	2	0.75	
Movable Type			Project occurred 8 years ago; estimate of staff time unknown.
obento (our in-house developed bento search)	4	500 (approx.)	
Open Journal System (OJS)	3	100	
Omeka	1.5	40	

OSS System	Staff	Staff hours	Comments
Omeka	2	60	Developer created an accessible fork of Omeka, called Omeka_a11y, for use in our library, then removed institution-specific changes and released the fork on GitHub.
Omeka	3	20	
Omeka	5	450	
Omeka		301	One digital exhibit
Open Journal System (OJS)	2	50	
Open Journal System (OJS)	2	400	
Papyrus	2	210	
phpScheduleIT	4	400	
ResourceSpace	1	8	
Room Booking	2	60	
RUanalytic	3	400	
Shibboleth	N/A	N/A	The development was driven by the university's Middleware Group, so it is difficult to estimate library time on the project.
Social Feed Manager	2	40	
UCLA Mobile Web Framework	1	40	Software started at UCLA to create a framework to have web sites work well on a mobile device without having to create apps for devices.
Vireo	2	200	Times are grossly estimated for the last question.
Vireo	2	120	For ETD management
VIVO	4	100	Deployment was spread over several months.
VIVO	6	250	
VuFind	2	500	
WebCalendar	1		
WordPress	1		
WordPress	2	25–35	We were already using WordPress on a limited scale for blogs and some web pages, but recently fully adopted WordPress for our library web site. Hours are based only on the time to setup and configure a new web server environment and WordPress instance for the intended use. Time spent creating and adding content was in addition and significantly greater.
Xerxes	2	2 * 280 hours	

Additional Comment

We do not have a metric for this at this time because it is not useful to capture unless we are comparing two similar scoped systems (OSS vs. Vendor). Much also depends on the type of application and needs it presents: rebrand requirements, training requirements, configuration and sometimes development to utilize.

14. Please identify your most recently adopted OSS system **that is still in production**, and indicate how many staff and how many staff hours per month are required to maintain the system. An estimate of the number of hours is acceptable. N=56

OSS System	Staff	Staff hours per month	Comments
ArchivesSpace	5	15	We are still in the process of migrating from Archon to ArchivesSpace.
Archivists' Toolkit	1	100	
Blacklight	3	200	
Blacklight	4	300	The system, though deployed, is still under active development. We cannot separate development from support.
Blacklight			We cannot share cost related information at this time.
CORAL	1	2	
DAMS – Islandora, Fedora Commons	2	280	The number of staff hours includes more than maintenance because the system is continually being developed for use beyond the library, to the entire enterprise. The two staff are working full time on the system, migrating digital assets from other legacy and proprietary systems into the DAMS, implementing authentication, user-centered interface and navigation, writing bulk ingesters, creating testing scripts, distributed solutions, data preservation processes, etc.
Droid	2	200	
Drupal	1	20	
Drupal	2	30–40	
Drupal	2	75	Two staff members are involved with maintaining Drupal, but not full time. It adds up to about .5 FTE.
Drupal	5	125	Library web site
Drupal	5	100	
Drupal	3		Hours unknown
DSpace	1	2	
DSpace	1	5	
DSpace	2	32	We are not currently tracking maintenance time for OSS systems.
DSpace	2	120	
DSpace	2	10	One Sysadmin handling patches/updates/security and one Developer handling feature requests and fixes.
eReserves	2	250	This is a locally developed system that we don't open source currently.
Fedora Commons	3	80	
Fedora Commons	4	512	
File Analyzer	1	20	
Guide on the Side	1	<10	

OSS System	Staff	Staff hours per month	Comments
Hippo CMS	10	40	Includes maintenance and occasional upgrades; does not include development of new website features.
Hydra	1	60	By "in production," in this question, it appears to us you actually mean still in development prior to deployment or in the earliest stages of deployment.
Hydra	3	100	
ICA-AtoM	2	20	
Islandora	1	70	
Islandora	2		We can't calculate staff hours with any accuracy as we haven't been systematically keeping track.
Nagios	0.25	1	For this OSS component, there only requires minimal effort to maintain, just the application of system patches.
obento (our in-house developed bento search)	2	20	
Open Journal System (OJS)	1	10	
Open Journal System (OJS)	3	75	24 instances; customer support and updates to software
Omeka	—	—	One digital exhibit
Omeka	1	10	The active installation requires minimal work. We are in the midst of a version update, to replace the current production installation—that is a larger time commitment, but I view it as a "project" not "support".
Omeka	1	2	Most effort spent sporadically when software needs to be upgraded.
Omeka	1.5	2	Very difficult to give staff hours per month; depends very much on the release cycle for product and status of projects being implemented.
Omeka	3	10	
Open Journal System (OJS)	1	8	Hours/Staff do not include continued development time.
Open Journal System (OJS)	2	44	
Open Journal System (OJS)	2	50	
RUanalytic	2	40	We are currently enhancing it via an NSF grant so spending more time on it than normal, particularly in response to feedback from grant P.I.
Shibboleth	N/A	N/A	This is incremental process, since we are supporting the university's single sign-on initiative. Library use of Shibboleth is being gradually phased in, with the goal of Shibboleth becoming the standard.
Social Feed Manager	1	2	
Solr/Nutch	3	20	Apache-based product to create a search index for our public web site.

OSS System	Staff	Staff hours per month	Comments
Spiceworks	2	4	For this question, we are assuming that "in production" means systems that we are actually depending upon, as opposed to systems that we have installed but not started to actively use ("deployed"), as in the previous question.
Umlaut	2	2 * 21 hours	
Vireo	2	< 10	
Vireo2.2	4	10	
VIVO	1	10	
VIVO	3	180	
WordPress, Confluence, JIRA, Jenkins	1 to 2	20	
WordPress	1	25	
WordPress	1		We have one full-time webmaster who spends the majority of his time doing custom design, maintenance, etc. on our WordPress site, as well as many other library staff who spend smaller percentages of their time creating content (blog posts, web pages, etc.)
WordPress	2	15–20	This is time spent maintaining the web server and WordPress environments and does not include time spent maintaining web site content.

Additional Comment

We do not have figures for separating software only maintenance and support and again is not useful unless comparing to something similar that offers the same functions. Much of the software we develop does not have vendor alternatives and our requirements go beyond just what the software delivers.

COST OF ADOPTING OSS

15. Were you able to track the costs of the most recently adopted and deployed OSS system? N=71

Yes	10	14%
No	61	86%

If yes, please indicate the costs of adopting that OSS system, and briefly describe what expenses were covered (e.g., staff time, equipment, training, travel, etc.) N=10

Cost	Expenses Covered
$400	Server hosting agreement for VM with university central IT department; cost here doesn't include staff time.

Cost	Expenses Covered
$646,119.07 over 4 years (yearly average cost $161,529.76)	Staff (IT, Archival, Tech Services), 3rd party developers, Amazon cloud hosting & storage
$3,800	3800
Approximately $8,000	Staff time
$50,000	Consulting, hosting, staff time, training, travel
$17,000	Vendor installation and support, virtual server, travel. Other costs not tracked so not included.
$40,000	Staff development time - NSF grant budget
$45,500	Staff time
We cannot share cost related information at this time.	We cannot share cost related information at this time.
Approximately $200,000	Staff time, equipment

What was the source of the funds for adopting this OSS system? Check all that apply. N=70

Library's operating budget	69	99%
Grant(s)	6	9%
Parent institution	4	6%
Consortial budget(s)	4	6%
Gift(s)	1	1%
Other funding source(s)	3	4%

Please specify the other funding source(s). N=3

2014 expenses will be reduced by the Amazon cloud hosting, storage, and back-up costs ($130,034.16) because the university's central IST department will provide these services locally.

We are able to track project costs but our practice is not to track time spent to implement.

We have library staff working on this project, but we have not tracked their hours, since it is part of their day-to-day duties.

BENEFITS AND CHALLENGES OF ADOPTING OSS

16. Please briefly describe up to three benefits your library enjoys as a result of adopting OSS systems. N=65

A cost effective means to deploy business critical software and services. Ability to customize for internal uses. Ability to serve users of the digital library with software standards and standard interfaces.

A single system hosts many formats; still images, books, newspapers, audio, video and manages all associated files, derivatives, preservation data. The core system was further developed to meet specific local functional requirements of users without waiting for vendor releases. The system is scalable to millions of objects and can provide a single enterprise solution for the whole university.

Ability to contribute bug fixes and enhancements desired at our institution. Lower initial cost outlay. Control over support and maintenance costs.

Ability to customize/extend the software to meet local needs. Easier to evaluate/test/prototype different options. Staff experience gained from working with the source code.

Ability to have applications that better meet the library's needs. Accessibility and usability are usually better for library patrons. Inline with library values to support open access.

Ability to have solutions more customized to our and our users' needs. Ability to provide innovative services beyond the reach of commercial products. Reduced dependency on vendor changes in products and priorities.

Ability to modify or change software based on specific needs. Community-based support and knowledge availability. Reduced/eliminated licensing costs.

Ability to rapidly respond to local needs/issues. Ability to configure/customize service to local needs. Local knowledge of interoperability issues with other systems in use by institution.

Because we have a local software development shop, we can adjust OSS systems to meet our requirements, and have succeeded in deploying systems that we believe are superior to commercial systems. The quality of OSS systems is often very high. OSS systems can evolve rapidly in response to new ideas and trends.

Better engagement with the communities doing the work. Ability to contribute to the improvement of systems used by libraries and archives. Better able to recruit and maintain developers from a wider circle of practitioners.

Built for a specific need. Cost of licensing.

Can customize to fit our requirements. Broader base of software support.

Community of Support. Better understanding of the technology. Good exit strategy.

Configurable. Broad user base. Ease of use.

Control and customizability. Speed to adopt. Ability to participate in community and shape direction.

Control of functionality. Participation in community over roadmap. Flexibility of customization.

Control over customization and software direction. Less effort to support. Functionality meets our needs.

Control over discovery system. Ability to expand scope of discovery system. Unlinking back end from discovery.

Control over system features and design. Reduced time to fix issues or troubleshoot.

Creation of highly collaborative environments. Increased knowledge/skills. Having a foundation on which modifications can be made to address local needs.

Customization. Connection to current systems. Ownership of data.

Customization. Community participation.

Developing and adopting OSS affords us flexible, sustainable solutions that meet complex problems facing libraries, archives, and museums. Reduces risk by affording control over the solutions that meet our needs and control over when and how to use them. Staff are working on solutions that have impact beyond our institution, have a professional network, higher visibility of the work they do while the library can save in training, resourcing, and stop gap measures during staff shortages.

Flexibility. Reduced cost and purchasing wait time. Community support.

Flexibility. Low risk in the case of project failure, due to nature of projects chosen. Customizability.

Flexibility in responding to changing needs. Opportunities to look for added value enhancements to services. Engagement with a wider community of library developers.

Flexibility to customize. Licenses are cost effective. Software easy to require.

Freedom to use, study, copy, modify, and redistribute solutions that work for us. Rapid access to really good ideas by people who don't work here with us. Implied membership in development communities.

Functionality that meets our needs. Ability to integrate software into our infrastructure, and with other library and university systems. Professional development opportunities from participation in the community.

Functionality that was not present in affordable commercial software. Ability to customize to meet our needs. Ability to integrate with local software.

Greater control of implementation timeframes. Lower up front costs. More flexibility with regard to customization.

Greater flexibility. No similar vended tools. Ability to develop new tools as needed from the OSS system.

Having access to a wide network of support for a system. Participating in a large community of developers with library-centric OSS expertise. Having more control over features and interfaces.

Improved quality. Customizability. Cross application integration.

Integration with other library systems. Opportunity to test software with little investment; low-cost testing/adoption.

Involvement at the national/international level. Can move to another product with no contractual lock-in. Opportunity to improve the product.

It gives us greater control over the implementation. There can be greater interoperability with OSS systems. The cost is internal; it generally includes staff time and training.

Less staff time to modify and support OSS systems when compared to creating homegrown products. We have better control over OSS software and our data than we do with vended products. OSS communities tend to have vibrant and engaged members, which can be a good support resource.

Leverage adoption community support. Attract applied research funding for OSS projects. Align with institute mission to share knowledge.

Lower acquisition cost. Complete control over user experience and user privacy. Flexibility.

Lower cost. Customizability. More control.

Lower licensing and maintenance cost. Fast deployment. Functionality sharing.

Many choices available. Allows for quick prototyping. Ability to modify to environment.

More options to choose from than just those provided by commercial vendors. Can frequently implement without need of identifying and budgeting funds to purchase product. Can implement more quickly because there is no need to go through a complicated and time-consuming licensing process.

No purchase cost. Community support. Flexibility to modify.

No purchase price. More control.

Obtaining functionality that best meets our needs. Control and customizability. Community participation.

Opportunity to contribute code that meets not only our specialized needs, but those of other institutions. Opportunity for developer to join a community of developers (professional development). Reflects our commitment to the values/ mission of the university and library profession.

Opportunity to influence future directions. Opportunity to increase staff expertise through reviewing and extending OSS code. Opportunity to leverage work at other institutions and contribute back to product.

Out of the box, relatively quick to install. Robust development community. Customizable face.

Prototyping; ability to try before you buy the "free puppy." Ability to customize to meet our needs. No licensing fees.

Provide additional services to user community. Less expensive. Greater ability to customize.

Quality of software. Ability to customize. Lower cost.

Rapid prototyping/updating. Community support. Reduced cost.

Save on licensing costs. Ability to customize, integrate with other library system. Research and publishing opportunities.

Shared expertise with other libraries. Customizability. Extensibility.

Software that is developed to meet the needs of the community rather than being profit motivated. Software that can be customized. Strong support community.

Speed of adoption. Services provided that would not otherwise be available. Good community support.

Staff development: increasing skill and knowledge. Flexibility in terms of being able to change without penalty. Rapid deployment: always faster to use OSS than a vendor solution for most anything.

Sustainability and influence in directing future development. More easily able to integrate other library platforms. Financial.

The ability to customize the product. The ability to influence the direction of development.

The ability to respond quickly and effectively to the needs of our user community. The ability to troubleshoot our systems because of the deep understanding we have of the software. OSS developer communities are more responsive than most vendors' support systems (at least in our experiences).

Tools and services that are designed and customized to real faculty and student workflow needs. Tools and services that integrate into a coherent and cohesive cyberinfrastructure. Reusable code that can enable building other things.

Using WordPress instead of our parent institution's commercial content management system allows us to develop a web site that is more attractive, more customizable, and meets our needs.

We have the ability to do deep customization without waiting for a vendor. We keep fixed costs down by avoiding proprietary licensing and support fees. We help improve the library OSS ecosystem by sharing our code and reusing other code.

17. Please briefly describe up to three challenges your library encountered as a result of adopting an OSS system and the strategies employed to overcome these challenges. N=64

Adapting the service for multiple users has been a challenge; we've addressed it by assessing user needs and conducting training. Systems security is a concern. We've addressed it through the use of penetration testing.

Adopting open source software isn't free. There are support costs. We schedule regular maintenance of our software. Some vendors have more resources and can be quicker to market to meet a need or respond to changing environment. To deal with this, we always keep our options open to swapping pieces between OSS and vended solutions.

Although we try to minimize support costs through good engineering, we nevertheless have to support the applications. We move most application support to a support group after deployment, but some support issues require developer attention, taking time away from development efforts on other projects. The time to deployment can be long depending on the level of development or customization we undertake.

Bad software. Bad documentation. Too much staff time needed to get application running.

Bugs

Change in mindset on part of technical staff to contribute to open source communities.

Changing code: careful tracking of changes. Pressure to always provide latest version: lots of testing.

Compatibility. Waiting for developers to make/implement fixes. Staff support.

Complex environment: use virtualized environment. Poor documentation: staff enhance documentation through various means. Rapid change: each successive version of a software is not necessarily implemented; assessed to determine the added value.

Configuration and customization may take time and may not be possible to customize to satisfaction. Idiosyncratic code which will need to be documented and systemized. Attitude that open source may mean an inferior product.

Continued maintenance. Documentation.

Coordinating activities across developers not in the same location. Managing expectations for features and delivery dates. Finding qualified developers and keeping them in the library.

Creation of new tools needs deeper understanding of the OSS system.

Customizability and time to maintain customizations. Resource time to support users in using as the software is somewhat unintuitive.

Deciding whether to develop custom extensions or install existing. Resolved through cost benefit analysis.

Difficulty in getting timely accurate support. Requires developing in-house deep understanding to support. Finding clearly written documentation. Building a documentation system to accompany OSS systems necessary. Understanding limitations in the feature set of an application. Building prototypes and involving stakeholders in pre-production testing.

Difficulty with interoperability. More staff overhead for maintenance and support. Unclear migration path.

Documentation. Adoption.

Documentation: develop local documentation; contribute testing, bug reports, and documentation to project. Incomplete functionality: develop alternative workflows, contribute enhancements. Poorly developed or managed code contribution process: minimize customization of software.

Ensuring enough cross training, especially to ensure continuity in case of staff loss. Handling non-core customizations in upgrades of core. Occasional gaps in documentation of OSS systems.

Finding and selecting products with the appropriate functionality. Discovery committees are usually tasked with the assessment and evaluation process. Conveying support knowledge from an experienced staff member to an

inexperienced staff member. In-house modifications to the OSS software can make this more challenging. The strategy for overcoming this challenge is to make extensive comments within the changed coding.

Gap in web design skills. Had to use existing resources. Difficult to organize functional teams to create requirements or user-stories. Developers filled gaps. Lack of a mature service model to offer support.

Having the skill sets to support the product over the long term. Having a voice in governance within the open source community. Software bugs with little or no support to fix issues. To overcome, we try to purchase vendor/3rd party support.

Highly skilled in-house staff required in lieu of vendor support. Deep customizations can create a local fork that is hard to upgrade for a new upstream release. The power to customize is addicting. Sometimes it's better to adjust the local workflow to fit a 90% good enough tool than to spend time building that last 10%.

Immature technology: chose only established and mainstream product. Lack of support: chose only product with available paid support. Lack of control on product and feature direction.

Increased deployment time for unfamiliar products: admins must spend more time learning software upfront. Users expect sys admins to be source of expertise for deployed products: have to educate users about becoming self-servant with available documentation and knowledge bases. Alignment of local project timelines with those of OSS products.

Initial hardware needs: repurposed hardware from other project. Reliance on locally developed expertise: limit the amount of customization.

Institutional IT department has had difficulty supporting large data, bandwidth, and open source philosophy in general. Core system needed considerable development beyond basic functions. Version updates not always scheduled or based on an upgrade path. Poor implementation and documentation.

It still creates IT debt that we need to manage. The communities are not big enough to always add value. We have a greater need for technical documentation when we release an OSS software.

Keeping up with software updates. Training overhead for new staff.

Lack of documentation: communication on listservs and forums.

Lack of documentation and support can slow adoption. Sustainability problems can lead to abandoned projects. Skepticism on part of non-technical stakeholders.

Lack of necessary elements: have developed our own or contributed to community work to do same. Lack of documentation.

Lack of staffing: we haven't really resolved this. Lack of training in specific areas: fortunately our location between two large metropolitan areas has made this fairly easy to obtain. Lack of policies and procedures for OSS: we have established a work team and are starting to address this.

Learning curve. Staff time. Server capacity.

Learning curve: overcome by online training resources. Recovering from patches to customized software: overcome by before/after detailed checklists. Training and maintenance: overcome by building in new routine tasks for maintenance and cutting back on other services.

Maintain thorough documentation of local implementation & customization decisions. Failsafe upgrades: need to make sure locally developed plugins, etc. don't crash with each new upgrade. Maintain sandbox environment to thoroughly test upgrades before pushing to production. Version control of development vs. production servers.

Managing all the associated software components of a software package. Getting the organization to make the appropriate level of investments. Free software does not mean no cost. Have to monitor security patches more closely.

Metrics that can be used to compare against commercial software since much of what we develop and use is done by OSS communities. We are not merely shopping, adopting, and tailoring; we are building it together and have no access to all the information needed for valid metrics. Strategy: gather information on cost for solutions that only serve a portion of needs and be able to articulate that against ballpark expense of equivalent OSS. Getting software developers from commercial sector to understand that the return on investment for day-to-day work is not exact. When you preserve cultural heritage or the scholarly record, the impact on research or learning is very difficult to measure; there is no clear profit margin in terms of money. Strategy: make applicants aware of the mission and strategy of the organization, be transparent about the institution and how the organization fits within the institution and the larger educational community. Managing expectations: since we have OSS, people believe they can have everything but we aim to standardize practices within our national and international communities so we have to manage expectations on how much customization and one-off design is sustainable and practical. Strategy: engage early, often, and be transparent into why and how work is being accomplished.

More complexity in implementation, configuration. Accommodating local customizations at time of software upgrade.

More up-front development work: it's all our responsibility. "Forking" code: ending up with code that is removed from the open source core.

Need to grow staff expertise: grew it.

New development method (agile) employed. Managing scope. Prioritizing desired enhancements.

Newer versions no longer supporting important features: overcome by changing to a different system. Minimal to no support: overcome by increasing our knowledge and expertise, or securing third-party support where available. Lack of availability of formal training in system use: overcome by taking a deep breath and figuring it out as we go.

Open source is not free. Infrastructure costs and developer salary/benefits add up over time. Keeping up with upgrades. Future of the product is not entirely up to us and may go in an undesired direction.

Personnel to sustain systems: proposal to administration to re-hire. Priority conflicts with multiple systems: working with leadership to implement portfolio management. No clarity on system expectations and service design when OSS solutions are requested from the IT department: working with leadership to implement project management.

Poor documentation for the software: our Systems Department was helpful getting the server ready, then we depended on an active and enthusiastic user group. Minimal tech support: we depended on fellow-users because help from the software was limited.

Problems must be resolved by staff: network with community of users. Documentation lacking: network with community of users; acquire reviews of OSS. Maintenance and upgrades: don't be the first.

Software ceasing to be developed by the community. Software being developed for technology stacks that we don't run. Inconsistent documentation.

Some software can have a steep learning curve.

Staff and consultant time spent on debugging and customization. Cost of implementation and support not much less than commercial product. Product looks behind-the-times.

Staff cost. Long-term stability and robustness of software. Open source licenses can be variable.

Staff time. Lack of support. Lack of clear documentation.

Support for changes, bug fixes is dependent upon user community. Future development can be taken in a different direction than desired, or stopped completely. Learning curve in the organization for production implementation & support after development. Not all open source software is documented well.

The main supporting group provides poor support or abandons the software. Dependence on technologies that are not well known within the library. Ability to both customize the system and track future releases.

Time to deploy. Compatibility among modules. Lack of documentation.

Total cost of ownership can be higher. Replacement of knowledge when staff involved in OSS project leaves. More difficult to justify investment in OSS over vended solution in face of budget cuts/constraints.

Transition plans for stranded (abandoned) OOS systems. In-house resources to support and extend OSS system hard to cultivate. Upgrade cycles are resource-intensive.

Trial and error approach is sometime necessary: need to have a tolerance for failure. Lack of community support at times. Development takes time.

Understanding features and capabilities of OSS now and in the future so we do requirements analysis and trial implementation. OSS can't be included as part of a formal RFP process: no strategy to overcome. Understanding the total cost of ownership for OSS: no strategy to overcome.

Unplanned costs associated with maintaining and customizing the code.

Variable level of support from the community, especially with older versions. Strategy: upgrade often! Sometimes missing one or two key features that are beyond the library's ability to develop in-house. Strategy: contract out to third parties. Greater staff time required to support. Strategy: ensure staff know the system thoroughly.

We locally customized one system and are a bit stuck with our fork now, but it's a tradeoff we manage just fine. Very good modern software tools often don't fit our legacy data; e.g., django requires utf8 db connections but Voyager requires us7ascii.

WordPress is not supported by our parent institution (university), so if we lost our in-library webmaster we would have no support.

LIBRARY CONTRIBUTIONS TO OSS PROJECTS

18. Has your library contributed to any library-related OSS projects (either your own or another organization's project) in any way (e.g., code or developer time, money, hosting)? N=72

Yes	56	78%
No	16	22%

If you answered Yes, you will continue to additional questions about your library's contributions to OSS projects.

If you answered No, you will skip to the section Additional Comments.

19. Please identify the open source software your library has contributed to. N=50

ArchivesSpace, Hydra

Avalon, Variations (testing partner)

Blacklight Reserves Direct OLE

Blacklight, Solr, Hydra, Vireo, Umlaut

Code for custom functions of our ILS

Developing a crowd-sourced transcription tool

Digital Preservation Network (DPN)

Droid, Pronom, storage Resource Baker, iRODS

Drupal, Citation Fox, IL Fox, Movable Type

Drupal, Omeka, DSpace, APTrust, Digital Library Extension Service (DLXS), Copyright Review Management System (CRMS), MPach, VuFind, Sakai, Solr, Lucene, Kaltura

DSpace (3 responses)

DSpace and File Analyzer

DSpace, Kuali, Fedora Commons, Hydra, django

DSpace, Silverstripe

DSpace, Vireo, CORAL

Evergreen, Islandora, Docker

eXtensible Text Framework (XTF). The work is in progress as of the end of February 2014.

EZProxy Wondertool, Mondo License Grinder, Archivematica

Fedora Commons (2 responses)

Fedora Commons, DuraSpace, ArchivesSpace

Fedora Commons, Blacklight, Hydra, Avalon, Hydramata, ArchivesSpace, APTrust, Digital Preservation Network (DPN), Solrmarc, Tracksys

Fedora Commons, Islandora

Guide on the Side

Hydra (2 responses)

Hydra, Blacklight

Hydra, CORAL, MyLibrary

Hydra, Blacklight, Umlaut, Xerxes, Drupal, ArchivesSpace, Archivists' Toolkit, Capistrano

In-house link tracking software, in-house map software, other contributions to VuFind

IR+. eXtensible Catalog, DSpace

Islandora, Archivematica, ICA-AtoM

KentDSS https://github.com/ksulibraries/KentDSS

Kuali Financial Systems, Shibboleth

Kuali OLE, Sobek, ASERL Disposition Database, jrnl

Kuali OLE, Avalon Media System, Fedora Commons, Hydra, Hydramata, Variations Digital Music Library, METS Navigator, Sakai

Kuali OLE, Global Open Knowledgebase (GOKb), LOCKSS, Solr, VIVO

LOCKSS (Private LOCKSS network)

Manakin (DSpace)

Manitobia, DSpace, ICA-AtoM, Islandora, Fedora Commons, LOCKSS, Drupal, Open Journal System (OJS)

Omeka

One example: Viewshare

SRA toolkit, BLAST, C++ toolkit, variety of scientific tools

SubjectsPlus, Remixing Archival Metatdata Project (RAMP), Variations Digital Music Library System, Avalon, Kuali OLE

There's a long list at https://github.com/gwu-libraries/

UCLA MWF, DSpace

VIVO, Fedora Commons

Voyager

20. Please indicate how your library is contributing to each of the following types of OSS projects. Check all that apply. N=56

Type of OSS Project	Code (i.e., developer time)	Money	Hosting	Other contribution	N/A	N
Institutional repository	32	18	5	10	14	52
Digital preservation	22	19	9	11	19	49
Digital asset management	20	8	4	5	26	48
Discovery layer	11	3	2	5	32	47
Publishing	5	5	5	3	34	47
ILS	6	5	—	7	37	46
Streaming media	7	4	2	3	37	46
Study room scheduler	5	—	—	1	39	45
Link resolver	3	1	1	1	41	45
Authentication/identity management	8	—	1	2	35	45
Inter-library loan	2	1	3	3	39	44
Data analysis	5	1	2	2	39	44

Type of OSS Project	Code (i.e., developer time)	Money	Hosting	Other contribution	N/A	N
Blogging	2	2	1	—	40	44
Electronic resource management	6	—	2	4	33	43
Course reserve	4	—	—	2	39	43
Floor maps	4	—	1	1	38	43
Data warehouse	6	—	2	1	37	43
ELMS	3	1	—	1	39	43
Visualization	4	1	1	2	39	43
Web analytics	3	—	1	1	38	43
Other type of project	15	5	2	6	16	30
Number of Responses	47	36	16	27	45	56

If you selected "Other contribution" above, please briefly describe the contribution the library makes to each corresponding project. N=25

Adding modules, patches as well as providing whole libraries (SRA toolkit, C++ toolkit, etc.)

Beta test institution

Blacklight: regularly host and organize committer calls; hosted Blacklight developer conference. Vireo: participate in the governance of the user community. Duraspace: Silver sponsors. Public Knowledge Project (PKP): Silver sponsors.

Both Kuali and Shibboleth are systems that are used university-wide. The Libraries is responsible for integrating these systems into our existing technology environment.

Consultation, organization

Contributing Omeka_a11y to the Omeka Project and ShadowPage, a page-turning plugin for content presentation in Omeka.

Contributing to and testing enhancements.

Creating software that intersects with OSS to enhance functionality.

Developing a crowd-sourced transcription tool.

Discovery layer, ILL, and "Other type of project": the library has contributed leadership, project management, governance, HR, financial management, and IT infrastructure support via the eXtensible Catalog Project, which developed four toolkits that fit within these various categories.

Feedback and bug reports for release candidates/new releases, contributing to support forms and listservs.

For both Citation Fox and IL Fox, library staff have provided training and given presentations at regional conferences.

Functional requirements, technical requirements, advisory role

Functional requirements, testing

ILS: project management, providing use cases. Electronic resource management: project management. Institutional repository: community membership.

Kuali OLE [ILS, ERM, Course Reserves]: participate to provide use cases, functional spec teams, testing of releases. Digital Music Library System, Avalon Media System [streaming media]: provide use cases, feedback on development priorities, release testing.

Legal advice, business/sustainability

Participation in architecture/design sessions, participation in pilot deployments

Release coordinator, educational efforts

Strategic direction, project management, research & development, grant management

Streaming media: bug reporting and testing (Kaltura). Digital preservation: we manage and offer fee-based support to this project.

Testing, feature requests/requirements development

We have a heavily customized VuFind instance. We share our changes on a publicly accessible source control server, but we're not pushing our changes up to mainstream VuFind (our customizations are too local-specific).

We have contributed to community engagement, hosted community meetings, facilitated planning teleconferences, and advanced the designs, strategic plan, and architecture of these projects.

We have participated in testing the Fedora Commons repository software.

If you selected "Other type of project" above, please briefly describe the project and the corresponding contribution the library makes. N=15

Archival management system: contributed to support forums/listers

Bibapp: Campus Research Gateway and Expert Finder

Citation Fox is open source software that organizes citations into four broad categories. IL Fox is open source software that provides users with tools related to information literacy.

Developing a crowd-sourced transcription tool.

Digital Humanities, Digital Scholarship tools

ICS-AtoM Archival records management system: code development, testing, feature requests/requirements

Omeka is an online exhibit building tool that the Libraries are using to support Digital Scholarship in the arts, humanities, and social sciences.

Scientific data analysis, text mining

Social media viewing/sharing and harvesting for archives: coding, project and community management

SubjectsPlus [research guides, FAQs, staff directory, database A-Z]: primary code development, documentation, distribution, support. RAMP [used to generate authority records for creators of archival collections (using EAC-CPF) and then take that structured data and transform it into wiki markup to facilitate the creation or enhancement of Wikipedia pages for those creators; also facilitates examination of names/organizations for quality control, data visualization]: development, distribution, support.

The eXtensible Catalog's Metadata Services Toolkit is a platform to transform library metadata into a variety of formats. The library contributed in all of the above areas to the development of this software.

VIVO: researcher profiles

We also contribute to a project called VecNet that isn't library related.

We are eliminating frames and developing the capability for responsive web interface design. We anticipate this to be included in the next version release of XTF.

Website content management system (Silverstripe) module

21. Please indicate how many OSS projects the library has contributed to and for how many projects your library was the *primary* code contributor. N=50

	Minimum	Maximum	Mean	Median	Std Dev
Projects	1	20	4.64	3.00	3.95
Primary Code Contributor	0	20	1.86	1.00	3.11

22. Please indicate how many library staff and about what percent of their time are dedicated to contributing to the development of OSS projects. N=46

	Minimum	Maximum	Mean	Median	Std Dev
Staff	1	14	3.89	2.00	3.34
% of Time	0.05	90	30.67	25.00	25.61

Number of Library Staff	% of Time
1	0.05
1	3
1	5
1	5
1	5
1	10
1	10
1	25
1	30
1	50
1	50
1	60
2	3
2	5
2	5
2	10
2	10
2	20

Number of Library Staff	% of Time
2	25
2	25
2	25
2	50
2	50
2	80
3	10
3	20
3	50
3	90
4	5
4	25
4	90
5	10
5	50
5	50
5	55
6	4
6	25
7	50
8	10
8	15
8	80
10	20
10	50
10	60
12	varies
14	50

LIBRARY AS ORIGINAL DEVELOPER OF OSS PROJECTS

23. Is your library the original developer for any of the OSS project(s) in which you participate? N=56

Yes	32	57%
No	24	43%

If yes, please identify the software. N=31

Archivists' Toolkit, ArchivesSpace

Avalon Media System

Avalon Media System, Variations Digital Music Library, METS Navigator

Blacklight for displaying complex digital objects, Oral History Management Software

BLAST, C++ toolkit, SRA toolkit, PubReader

Citation Fox, IL Fox

Co-primary developer of Fedora Commons 4

Curator's Workbench

Custom Voyager Reports Server

Developing a crowd-sourced transcription tool

Discovery: a Solr-based discovery tool that generalizes an index, search, browse, and deliver framework that can work with content such as MARC records or EAD finding aids, but also including non-library context such as open access publication of scholarly research, and a working catalog of global language observations by an international community of scholars.

Digital Library Extension Service (DLXS)

DSpace

ETD-db, ETD-db 2.0

EZProxy Wondertool, Mondo License Grinder

Guide on the Side

https://github.com/ksulibraries/KentDSS

Hydra (parts of), CORAL, MyLibrary, VecNet

In coordination UVa with Cornell: Fedora Commons; in coordination UVa with Stanford and University of Hull: Hydra; UVa: Blacklight; UVa: Solrmarc; UVa:Tracksys; in coordination UVa with Roy Rosenzweig Center for History and New Media: Neatline.

IR+. eXtensible Catalog

RAMP SubjectsPlus

See https://github.com/gwu-libraries

Simple Archive Format Packager: a tool to support batch ingest of content into the institutional repository (DSpace) (in Java)

Sobek, ASERL Disposition Database, jrnl

Sufia (a Hydra-based repository application)

Suma (mobile space assessment toolkit), lentil (Instagram viewing/sharing, and harvesting for archives), djatoka Ruby gem (Image server wrapper)

Umlaut was originally developed by Ross Singer. We took it over very early on and have been the principal developers since. Our library is the primary developer for the Data Conservancy.

Viewshare is the LC instance of the Recollection OSS software, so not totally created ab novo at LC but considered an LC product now.

Vireo, Collaborative Book Reader (CoBRe)

VuFind, Papyrus, Isladora

We created link-tracking software and map software that is OSS but currently only in small release (code shared upon request). We plan to clean up these projects (and several others) to move them to a public GitHub repository.

Please indicate how important each of the following reasons for deciding to open source the project is to your library. Please make one selection per row. N=43

Reasons	1 Not Important	2	3	4	5 Very Important	N
Shared effort in development and quality assurance of the product	4	5	7	13	14	43
A desire to contribute to an open source community	1	3	10	15	14	43
A belief that open sourcing would lead to better software	1	6	5	17	13	42
A need for expertise not available in your institution	11	9	11	6	4	41
At the request of another institution	14	7	12	6	2	41
Other reason(s)	2	—	1	3	6	12
Number of Responses	22	23	29	31	31	43

If you indicated above that the library has other reason(s) for deciding to open source the project, please briefly describe the reason(s). N=10

Ability for others to adapt tools to meet their needs. Provide support for platforms and services that are not required by our institution.

Assistance with ongoing sustainability of the product.

Demonstrate expertise of library staff to project in a non-library context. Develop an alternative business to deepen the libraries' engagement with researchers and scholars.

How good the system is.

Need for tools not otherwise available.

Other libraries have shared generously before us. We have the expertise and feel some duty to share alike.

Requirements of granting agencies that software developed with grant funds be shared under an open source license.

Risk reduction with resourcing, sustainability and exit strategy.

There was nothing available at the time that ETD-db was developed. Its recent rewrite was entirely for the external use community.

Training aid, set an example

COST OF CONTRIBUTING TO OSS PROJECTS

24. Were you able to track the costs of your most recent contribution to an OSS project? N=53

Yes	10	19%
No	43	81%

If yes, please identify the most recent OSS project, indicate the costs of contributing to that project, and briefly describe what expenses were covered (e.g., staff time, equipment, training, travel, etc.) N=10

OSS Project	Costs	Expenses Covered
Avalon Media System	Not available	Travel to meetings and conferences
Crowd-sourced transcription tool	$7500	Consultant, in-house staff time
Custom Voyager Reports Server	Staff time and equipment	Staff time and equipment
DSpace REST API	Approx. $10,000	Salary/benefits (2 months developer time)
Fedora Commons 4	Pending	Pending
Fedora Commons	We cannot share cost information at this time.	We cannot share cost information at this time.
Open Journal System (OJS)	5% of developer time	Staff time, travel
Open Journal System (OJS)	$2750	Conduct design work, client meetings, programming, testing, troubleshooting, and documentation
Papyrus	N/A	Staff time
Vireo	1 FTE for 1 year	Wages, travel, training

What was the source of the funds for contributing to this OSS project? Check all that apply. N=45

Library's operating budget	43	96%
Grant(s)	10	22%
Parent institution	3	7%
Consortial budget(s)	2	4%
Gift(s)	1	2%
Other funding source(s)	2	4%

Please specify the other funding source(s). N=2

Able to track, chose not to track. Would come from library's operating budget.

Funded by another university division (Technology Services)

BENEFITS AND CHALLENGES OF CONTRIBUTING TO OSS PROJECTS

25. Please briefly describe up to three benefits your library enjoys as a result of contributing to OSS projects. N=44

Ability to enhance product and influence its direction. Sharing with community.

Ability to influence project outcome.

Ability to lend expertise to peer or smaller institutions. Mutual benefit from reusing working solutions.

Avoids data lock-in. While it may not be any less expensive/time consuming to migrate data out of an open source system than a proprietary system, at least with open source there will always be the technical possibility. User communities and developer communications tend to be better formed, enabling better DIY support, and not being totally reliant on a single vendor. Open source values (access to and right to share information) map closely to library values.

Becoming an active part of worthwhile communities. Helping make products we and others use better. Increase our skills and expertise and inspire productive creativity.

Better service offerings. Alignment with institute mission. Collaboration with non-library departments and peer institutions.

Broadens their perspective as developers, product owners, and project managers. Meets the strategic needs of the organization to engage with the world and our communities. Helps us build better solutions with like-minded people and institutions.

Collaborating with other institutions to address common areas of need. Involvement of library staff in intellectually engaging and useful work. Ending up with a more sustainable product than if we had done it just on our own.

Collaboration of common tasks. Faster return on requested features. Giving back.

Community is able to benefit from our developments. Forces us to write cleaner code that is generalizable and fits with our strategies for replaceable parts.

Contributing code helps to meet our specialized needs. We participate in a community of experts. Contributing to the project is in accordance with the Libraries' and university's mission.

Contributing to the library community. Developing local expertise. Recognition.

Contributing, even in a small way, to non-commercial, inexpensive, and highly functional alternatives to expensive commercial software that drain our budgets. Good press for the university, and for the Libraries. Providing software to fill needs of other institutions.

Control of product design. Functionality meets our needs.

Credibility in OSS Developer community. Ability to share problems. Modeling good behavior.

Customization for our exact needs.

Enhanced quality of software through collaboration. Leveraging effort from multiple institutions. Ability to use work from other organizations.

Ensures product remains stable and useful. Fulfill our obligation as a user of the OSS. Improved understanding of the OSS.

Freedom to use, study, copy, modify, redistribute our solutions. Participation in a broader community. Visibility in that community as a contributor.

Functionality that best meets our needs is built into the software. Community participation. Identification and reporting of bugs and new features.

Gain respect as industry leader. Community enrichment. Education.

Good library citizens/community contribution. Having features released that we require. Exposure to new ideas and professional learning and sharing from a broader community.

Increased visibility. Added enhancements.

Institutional needs more likely to be accommodated.

Institutional recognition. Creating a better product than what was currently available. Opportunities for collaboration both within the US and abroad.

Latest software releases. Ability to help steer direction of software development. Ability to tailor software to local needs.

Our monetary contribution helps to sustain the open source federation.

Prestige. Providing direction. Collegial atmosphere.

Pride. Forces rigor.

Providing flexible solutions to solve common library issues or service requirements. Professional development of team members & providing exciting/challenging work environment.

Recognition. Control of budget.

Recognition and community building. Opportunity to influence product development.

Recognition as a source of expertise. Input into direction of software development.

Reduced support costs: others can adapt tools rather than requesting us to make changes. Ability for others to enhance and expand on previous efforts.

Safety in numbers: use helps to ensure viability of the solution. Revenue from offering support. Bug reports and occasional code contributions.

Shared development

Staff development. Reputation. Collaboration building.

Sustainable solutions: together we go farther. Sum is greater than the parts: quality solutions that meet our needs. Investment in our staff: more meaningful work, deepening skills, end of isolation.

Tool is available to meet our needs. Customizability. Ability to add features as needed.

Visibility and participation in the community. Investments benefit other libraries and can lead to partnerships, other collaboration.

We are part of the OSS community.

We helped the Avalon and Variations projects through testing.

We use software to solve our problems that others have written. Better code is written when you have an external audience of coders reviewing your contribution. There's lots of it that's relevant to an academic library.

We want to be able to influence the direction of the effort to align it with our needs. By participating in a larger community, we can contribute the good ideas of our staff and in turn learn from the good ideas of others.

26. Please briefly describe up to three challenges your library encountered as a result of contributing to OSS projects and the strategies employed to overcome these challenges. N=37

Adhering to community standards that differ from in-house. Committing the resources to develop contributions. Understanding the code base and requirements according to the community need.

Agreement of product direction. Coordinate development.

Assessing value to OSS project. Confidence in coding standards. Compliance with OSS review process.

Contribution of developer time can compete with other local project priorities. Remote/asynchronous collaboration: might have to wait a long time for responses. No clear, quantifiable ROI.

Coordinating effort across institutions challenging/varying opinions on functionality. Finding financial sources. Maintaining and supporting software.

Coordination/management of developers. Getting good functional requirements.

Developer/programmer will graduate. Staff required to learn programming of system. Need to document every phase.

Developing a product that is generic enough to meet needs of multiple institutions. Supporting and growing the community around the project. Sustainability: securing ongoing funding to support the software.

Difficult to make substantial contribution without more dedicated time to devote to it.

Extra time. Convincing stakeholders of value. Coming to terms with applicable licensing models.

Finding staff time to contribute. Disconnect between OSS priorities, which may be based on the funder's priorities, and our institutional needs. Ongoing financial commitment as OSS moves to a community source model.

Finding time and resources to devote to development process. Feature creep.

Finding time to contribute. Time to support and answer questions. Removing localization.

Getting library staff familiar with OSS/collaborative ways of working. Lack of control of timelines of collaborative OSS projects, need to readjust expectations. Not enough staff time to both participate actively in OSS projects and continue local responsibilities.

Increased time spent in detailed documentation.

Internal buy-in to benefit of time spent on OSS projects: communication about project at all levels of institution; reaching out to potential stakeholders early in process. General Consul was concerned about our distribution of code, especially with development contributed by faculty who don't have code development built into their job description. The faculty had to sign a release before we could contribute the code.

It can take more work to contribute well to a public project, but that can tend to produce better results. We need to review legal guidelines around assigning copyright to external organizations.

It is more expensive to write code that is generalizable than custom code for your institution. The development process is slower and requires a higher mind.

Larger than expected contribution time required of local resources.

Legal and licensing issues. Strategy: involvement of in-house legal expertise (our Director of Copyright and Digital Scholarship) and coordination with the university Technology Transfer office. Need to provide support or decide how much support to provide. Strategy: clearly communicate expectations regarding level of support provided. Need to support a wider range of environments than would be necessary for an internal-only deployment. Strategy: reducing over-dependence on current architecture can actually reduce costs over the full life of a project.

Maintenance of contributed code to fill the needs of the outside community. Monitoring feedback through multiple channels (pull requests, forum posts, IRC, etc.)

Managing expectations, sometimes you have to compromise. Strategy: engage with people and be transparent. Determining which projects to engage and to what degree. Strategy: stay connected at a management level, know your strategic objectives, know your staff and what culture is a good fit for your resources. Resources. Strategy: be able to show value toward strategic objectives for the resource investment.

Meeting expectations of adopters when we are the primary contributors.

More meetings take time away from local development.

Not having solid business models to refer to showing the real costs of developing, supporting, using OSS. Not being able to devote enough staff effort to OSS projects. When they are on a project less than 50% their return on investment is not as great. Getting institutional support beyond the library for certain solutions. Many administrators seem to prefer vendor provided out-of-the-box solutions.

Opportunity cost: developers not able to contribute to local initiatives.

Partner reliability.

Product was too narrowly focused for our exact needs to be worthy of sharing out to the community.

Some open source applications don't have formal paid support options available, so support risks are transferred from a vendor to the institution: careful evaluation of the risk, and level of risk before making the decision to do an OSS project. Sometimes a lack of understanding that open source doesn't equal free. The cost to the institution may be the same or even greater than a proprietary solution, just the money is spent on different aspects of the project: discussions with library stakeholders to make sure everyone clearly understands the full cost of OSS projects. Lack of institutional understanding to the open source model and licenses can hinder contributions of code back to the community.

Staff time: we just juggle this part with regular projects.

Support requests related to OSS projects takes some time.

Time and effort for creating it. Maintenance.

Time and resource commitment

Time spent to keep track of project.

Time to develop: fit in around other responsibilities. Time to support/answer questions: make part of professional development responsibilities.

Time: overcome only by choosing not to move forward on other projects at that time.

Uses valuable staff time: overcome by making sure we only contribute time we can afford and/or that will provide a desirable return on investment.

TOOLS FOR OSS PROJECTS

27. Does your library use a public repository or forge (e.g., Github, Sourceforge, Google Code, Bitbucket) to share your open source code? N=52

Yes	41	79%
No	11	21%

If yes, please identify the repository or forge. N=41

Github	38
Google Code	3
SourceForge	3
Bitbucket	2
Drupal GIT	1
RedMine	1
Subversion	1

Comments

Currently not, but we're moving to Github.

We're exploring doing this in a more standardized, regular way, but are exploring security concerns.

28. What tools does your library use to facilitate collaboration on the OSS projects your library contributes to? Check all that apply. N=45

Shared version control	37	82%
An issue tracking software package	36	80%
A mailing list	32	71%
A wiki	25	56%
A forum	12	27%
Other tool(s)	10	22%

Please briefly describe the other tool(s) your library uses to facilitate collaboration on OSS projects. N=10

Conference calls

Google Docs

IRC

IRC for chat collaboration

IRC, Google Hangouts, Adobe Connect, Skype

PivotalTracker

Project management tools (e.g., Trello)

Skype

Trello

Virtual tools for the team, project management software

LICENSING MODEL FOR DISTRIBUTION OF OSS

29. What licensing models does your organization recommend for distribution of software? Check all that apply. N=42

GNU Public License (GPL) version 3	16	38.1%
Apache	15	35.7%
Creative commons	15	35.7%
MIT	12	28.6%
GNU Public License (GPL) version 2	11	26.2%
BSD 3 Clause	3	7.1%
BSD 2 Clause	2	4.8%
Other licensing model	12	28.6%

Please briefly describe the other licensing model. N=12

Educational Community License (ECL) - ECL 2 (2 responses)

Educational Community License (ECL)

I wouldn't say that we've come across this very often or that we have a strong opinion of which licenses to recommend. If asked, I'd recommend that we evaluate these options and use the license that best fits the software. Much of the code we write falls under the license used by the platform or libraries that we leverage. Further, we haven't really been open sourcing any internally developed applications.

Internally developed Rights Statement based very closely on CC.

OSS produced at LC is generally considered federal work product and public domain.

Public domain

Public domain (Creative Commons - CC 0)

There is no organizational policy on licensing models.

This is just what we've used; there is no standard license that we would necessarily recommend.

We don't recommend it per se, rather we use an MIT-style license on our own software, as approved by the university.

We have no formal recommendation.

OSS PROJECT ASSESSMENT

30. Please indicate how important each of the following indicators that your contribution to an OSS project has been successful is to your library. Please make one selection per row. N=51

Reasons	1 Not Important	2	3	4	5 Very Important	N
The functionality better suits our institution's needs	—	—	1	8	41	50
Amount of community contribution/involvement	1	8	14	17	10	50
Number of project adopters	2	8	15	18	7	50
Number of project releases	4	11	23	9	3	50
Ease of support	—	2	21	15	11	49
Staff time savings	5	7	17	14	6	49
Monetary savings	4	13	10	17	5	49
Other indicator(s)	2	—	1	1	1	5
Number of Responses	11	22	45	40	46	51

If you indicated above that the library relies on other indicator(s) that your contribution to an OSS project has been successful, please briefly describe the indicator(s). N=3

Community interest in project [altmetrics, conference presentations, articles]

We are concerned to ensure that software systems are section 508 compliant, this indicator of success is not necessarily subsumed under "functionality."

Sustainability in terms of direction and responsiveness to meet evolving needs.

Additional Comments

Again, we don't agree that OSS results in staff time savings or ease of support, so did not respond to those two statements.

Who has adopted, and not just the number of adopters.

LIBRARY DOESN'T USE OSS

31. Please briefly describe why your library is not using any open source software. N=2

We don't have a sufficient IT support to develop, customize, and maintain OSS software.

We have not done any major software selection processes in over five years, and the OSS products have not historically had the functions we required. That may be changing looking forward.

ADDITIONAL COMMENTS

32. Please enter any additional information that may assist the survey authors' understanding of your library's use of open source software. N=19

I forgot to add that we developed a collection directory application, currently used for two projects, WAAND (Women Artists Archives National Directory) and NAP (Newark Archives Project).

Last August we hired a programmer with Drupal skills to assist in the library's web site redesign. We are trying to get colleagues to use Gimp because the licensing fees for Adobe Photoshop are prohibitive. Needless to say, Gimp is not being well received, yet. The campus and university system procurement office is trying to negotiate a campus and system-wide license.

OSS allows for greater customizations that fulfill the needs of so many library patrons and employees. We are lucky enough to have enough staff to get started on these projects, but it was very important for us to agree on some core OSS elements to make it easier to maintain in the long run. A good example of this our use of PHP and Apache. Focusing on this as a core allows for a smaller number of programmers to turn out and support a large number of applications. I will note that we have a smaller use for MySQL as there is a significant cost reduction in licensing Microsoft SQL for the university system. Therefore, we are not in the norm in that our Linux, PHP, and Apache works more with Microsoft SQL than MySQL.

OSS is a cost effective way to provide solutions that can be customized to local needs. The various components can be used to build products and solutions large and small. A staff of skilled software developers is required to use the tools, and products. It also requires system support staff to learn and support new tools, especially database systems.

OSS is used to support operations. Currently, not a major focus. Generally not using because of development and maintenance costs (staff time).

The availability of staff skilled in OSS technology remains the one hurdle to implementing more OSS as a strategy for the library. There is great interest in utilizing OSS more widely as a part of our technology strategy, but balancing availability of skillsets vs. demand will be challenging.

The Libraries and Academic Computing and Networking Services (ACNS) both report to the Vice President for Information Technology/Dean of Libraries.

The library has the will to participate in OSS if we had the staff time and resources to commit to OSS projects.

The use of OSS is very important to our mission, resource, and risk management.

This survey didn't ask about future projections of OSS use. We currently have DSpace but are devoting several full-time staff to developing Fedora Commons and Hydra. IT staff are divided between the ITS department and the Center for Digital Research and Scholarship.

We are a typical large research university. The use of OSS for interface to the digital library (REST APIs) allow for our research faculty to create content with whatever tools they are comfortable with. We encourage use of our standards, but if they use the API, they can do what they please with our digital assets.

We are very supportive of OSS but ultimately use the products that best meet our needs. Sometimes this is OSS but sometimes it is a commercial vendor product as there are advantages and disadvantages to both.

We believe in it deeply. It's what we do. We'd be up a creek without it.

We have no preference for OSS over vendor software. We use what works best and what we can afford.

We learned (the hard way) from our first experience with putting OSS developed elsewhere into production (about 10 years ago) that having vendor support and an active community around an OSS application are very important. With the OSS that we have developed locally (eXtensible Catalog and IR+), we have been unable to provide either of these things to potential users of our software, and have thus found ourselves in this same position with our own software of being unable to sustain the software on our own. While we still strongly support OSS and continue to implement additional OSS applications, we now make sure that vendor support and an active user community are already in place before we proceed with deploying the software.

We take a broad view of OSS and answered based on that approach, not limiting the scope to library-specific OSS. Our answers would be different were this more clearly defined, perhaps. Also, it suffices to say that our philosophy is simple: open source first, vendor only when there's no viable OS option. For example, we run our own data centre, and for that infrastructure from operating system to virtualization platform, it is all OS; there's no VMware, Citrix, etc.

We're transitioning from using mostly closed software to preferring mostly open software, so we're not yet where we want to be. We're working out more formal policies with campus technology transfer to allow us to release GPL software at our own discretion. We choose to use more OSS than vendor software because we have a tight budget but a great IT staff. With much of our software support burden being internal, it doesn't leave a lot of time to take the extra steps to polish, release, and support OSS software. But it's still a major goal for us.

While we use OSS, our unwritten policy is to use hosted, out-of-the-box solutions wherever possible. OSS is used to fill in the gaps.

RESPONDING INSTITUTIONS

Arizona State University
Auburn University
Boston Public Library
Boston University
Brigham Young University
University of British Columbia
Brown University
University of Calgary
University of California, Irvine
University of California, Los Angeles
Case Western Reserve University
University of Colorado at Boulder
Colorado State University
Columbia University
University of Connecticut
Duke University
University of Florida
George Washington University
Georgetown University
Georgia Institute of Technology
University of Guelph
University of Hawaii at Manoa
Indiana University Bloomington
Iowa State University
Johns Hopkins University
University of Kansas
Kent State University
University of Kentucky
Library of Congress
University of Louisville
McMaster University
University of Manitoba
University of Maryland
Massachusetts Institute of Technology
University of Miami
University of Michigan
National Archives and Records Administration
National Library of Medicine
University of Nebraska–Lincoln
University of New Mexico
New York University
University of North Carolina at Chapel Hill
North Carolina State University
Northwestern University
University of Notre Dame
Ohio University
Ohio State University
University of Oklahoma
Oklahoma State University
University of Oregon
University of Ottawa
University of Pennsylvania
Pennsylvania State University
Purdue University
Queen's University
University of Rochester
Rutgers University
University of Saskatchewan
University of Southern California
Southern Illinois University Carbondale
University at Albany, SUNY
Syracuse University
Temple University
University of Tennessee
Texas A&M University
Texas Tech University
Tulane University
University of Utah
Vanderbilt University
University of Virginia
Virginia Tech
Washington State University
Washington University in St. Louis
Western University
University of Wisconsin–Madison
York University