Barriers to Initiation of Open Source Software Projects in Research Libraries

Jason Curtis Thacker
Brigham Young University - Provo

Barriers to Initiation of Open Source Software Projects in Research Libraries

Jason Curtis Thacker

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

May 2016

ABSTRACT

Barriers to Initiation of Open Source Software Projects
in Research Libraries

Jason Curtis Thacker
Department of Computer Science, BYU
Master of Science

Libraries share a number of core values with the Open Source Software (OSS) movement, suggesting there should be a natural tendency toward library participation in OSS projects. However, Dale Askey's 2008 Code4Lib column entitled *We Love Open Source Software. No, You Can't Have Our Code*, claims that while libraries are strong proponents of OSS, they are unlikely to actually contribute to OSS projects. He identifies, but does not empirically substantiate, six barriers that he believes contribute to this apparent inconsistency.

The goal of this thesis is to empirically investigate not only Askey's central claim but also the six barriers he proposes. Additionally, we will utilize statistical methods and machine learning algorithms to identify barriers encountered by libraries as they grapple with whether or not to release their code as open source. We will offer insights into possible correlations between a library's engineering, talent management and innovation policies and practices and its propensity to initiate open source software projects.

Keywords: open source software, research libraries

ACKNOWLEDGMENTS

I acknowledge the support and love of my family. To my parents, Todd and Trudy, thank you for your love, council and encouragement. Thank you for being proud of me. To my children: Paige, Peter, Parker, Peyton, and Preston, you are a constant source of passion and motivation.

I express deep gratitude to Dr. Christophe Giraud-Carrier and Dr. Charles Knutson who have elevated my thinking and taught me to write. Certainly, this work would not be possible without their help.

From the Harold B. Lee Library, I acknowledge Bill Lund and Grant Laycock who have offered consistent encouragement and support. I also thank Scott Bertagnole who has been an editor and sounding board through this process.

Most of all, I thank my fantastic wife, Brianne, for her confidence, love and enduring patience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Background

1.1.1 Empirical Software Engineering

Empirical software engineering studies the process and people problems related to the creation of software. The formalization of empirical software engineering research came in 1986 when Vic Basili presented a framework for analyzing experimental work in this area [1]. Basili described empirical software engineering as a laboratory science [2]. In recent years researchers have applied techniques from sociology, psychology and data mining to inform this research process [3-5]. Sjøberg et al. [4] describes empirical methods this way:

> *Software systems form the foundation of the modern information society, and many of those systems are among the most complex things ever created. Software engineering (SE) is about developing, maintaining and managing high-quality software systems in a cost-effective and predictable way. SE research studies the real-world phenomena of SE and concerns (1) the development of new, or modification of existing, technologies (process models, methods, techniques, tools or languages) to support SE activities, and (2) the evaluation and comparison of the effect of using such technology in the often very complex interaction of individuals, teams, projects and organizations, and various types of task and software system. Sciences that study real-world phenomena, i.e., empirical sciences, of necessity use empirical methods, which use consists of gathering information on the basis of systematic observation and experiment, rather than deductive logic or mathematics. Hence, if SE research is to be scientific, it too must use empirical methods.*

Significant empirical software engineering research has been done in BYU's Software Engineering Quality: Observation, Insight, Analysis (SEQuOIA) lab.

1.1.2 Open Source Software

One of the major research areas within empirical software engineering is open source software (OSS). 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" [6]. The open nature of OSS gives easy access to source code, code repositories, contributors and other project data [7-9]. This data gives a unique view into the creation of software. Researchers have also studied many other aspects of OSS including motivations to contribute [10-15], barriers to adoption [16-18], the application of OSS principles in a professional setting [19], the structure of OSS projects [16, 20-22] and even the meaning of OSS [23, 24]. BYU's SEQuOIA lab has published on the subject of OSS many times in the last 6 years [9, 22, 23, 25-31] and has contributed sundry insights to the OSS community. In addition researchers have explored various contexts of OSS such as real-time applications [32-34], medicine [35-37], and education [38-40] to name a few.

1.1.3 Academic Libraries & Open Source Software

Libraries rely heavily on software to carry out their basic business functions. Much of this software is Commercial off the Shelf (COTS), however adoption of OSS is also becoming a viable option. There are many library specific open source software projects. The adoption of, contribution to and initiation of OSS projects in the Library IT context is only beginning to be studied.

The mission statement of the American Library Association includes the charge to "ensure access to information for all." This charge comes without cost or qualification. Stated another way, libraries make information freely available to all regardless of how that information will be used. The core values of libraries and the OSS movement are similar, suggesting that libraries should tend to favor the OSS model. In particular, they may feel a responsibility to share the code they have developed with other libraries in a spirit of openness and access for all.

The predisposition of libraries toward OSS adoption and contribution is not a new idea. Pat Eyler, an open source developer for the Koha ILS project, said “That more librarians aren’t actively using and evangelizing free software is an indictment against us for not letting them in on our secret” [41]. Richard Stallman, the pioneering free software evangelist, stated that “… universities shouldn’t be developing proprietary software. It is better if they develop none at all, because [by doing so] they are betraying their mission to contribute to human knowledge” [42]. Nicole Engard characterized the issue this way: “It has been suggested that libraries are almost ethically required to use, develop and support open source software” [43].

Despite the suggestion that libraries are ethically required to use and create OSS, it has been observed that libraries seem reluctant to share their code. In 2008 Dale Askey authored a paper entitled *We Love Open Source Software. No, You Can’t Have Our Code*. He states that “Librarians are among the strongest proponents of open source software. Paradoxically, libraries are also among the least likely to actively contribute their code to open source projects” [44]. Further, Askey identified a list of six likely interrelated issues that he believes contribute to this dichotomy. In his own words:

- perfectionism – unless the code is perfect, we don’t want anyone to see it.
- dependency – if we share this with you, you will never leave us alone.
- quirkiness – we’d gladly share, but we can’t since we’re so weird.
- redundancy – we think your project is neat, but we can do better.

- competitiveness – we want to be the acknowledged leader.
- misunderstanding – a fundamental inability to understand how an open source community works.

The validity and potential impact of these issues have not been tested empirically. In this thesis, I will create an instrument to empirically investigate Askey's central claim. I will also examine the six barriers he proposes in light of my empirical results. Further, I seek to identify the characteristics of libraries that initiate OSS projects.

1.2 Project Description

This thesis describes the creation of a survey to empirically test the prevalence of OSS adoption, contribution, and initiation practices, and is an exploration of these findings within the context of research libraries and open source software. Chapter 2, published as ARL SPEC Kit 340 [45], contains complete survey results, as well as the details of survey preparation and administration. Chapter 3, originally published in the code4lib journal [46] responds to Askey's claims. Chapter 4 is a yet to be published paper that outlines and discusses findings discovered utilizing an expanded dataset and data mining techniques. Finally, chapter 5 contains concluding remarks and future work.

Chapter 2

SPEC Kit 340: Open Source Software[1]

2.1 Executive Summary

2.1.1 Open Source Software

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 is 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 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

[1] This chapter is published as ARL SPEC Kit 340, 2014 [45].

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.

2.1.2 Library IT

This survey was distributed to 127 ARL member libraries in February 2014. Seventy-seven libraries (61%) responded to the survey.

For libraries affiliated with research universities, Library Information Technology (LIT) averaged 15.7 staff members, with a median of 14.0, minimum of 2, and maximum of 50. For governmental libraries (Library of Congress, National Archives and Records Administration, and the National Library of Medicine), library IT organizations were significantly larger, averaging 243.3 staff members, with a median of 250, minimum of 130, and maximum of 350. Only one public library was represented in the survey with an LIT organization of 30 staff members. The bimodal distribution of LIT organizations by staff size is stark, with governmental 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 of governmental libraries in OSS projects compared to research university libraries.

Seventy respondents (91%) developed software in-house. Of those, the most common software development practices included using version control (86%) and performing usability tests (86%). The least common practices included 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 software practices mentioned by respondents in the comments were agile/scrum development methodologies (5 respondents) and pair programming (2 respondents).

Most respondents reported that their LIT staff were encouraged to experiment with new technologies (99%), and prototype potential projects (82%).

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

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

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

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).

In our review of organizations that contribute to open source projects, software development staff ranged from one or two to as many as fourteen. 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 were often located in library systems or web units. Within larger organizations, software development staff were often clustered together in application development units located in digital library, digital projects, or library technology branches of the organization.

2.1.3 Adoption

Seventy-four (97%) respondents have deployed open source software in their library. Each respondent was asked to provide information about the type of software being used for various purposes. Below are some of the highlights.

- Fifty-eight respondents (76%) use a vended, locally hosted integrated library system (ILS). No respondents use an ILS built in house, but four use an open source ILS.
- Forty-five respondents (59%) use a vended, locally hosted interlibrary loan (ILL) system and twenty-nine (38%) license a software as a service (SaaS) ILL system.
- Forty-nine respondents (64%) use a SaaS discovery Layer. Seventeen respondents (22%) use a vended, locally hosted discovery layer, and ten respondents (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 institutional repository.
- Forty respondents (53%) use a locally hosted and supported digital preservation system.

- Thirty-four institutions (45%) 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 assent management systems (19 respondents).
- The systems most frequently adopted as open source systems include digital repositories (57 total), institutional repositories (54 total), blogging (53 total) and publishing (43 total).

Forty-three respondents (59%) had no formal library or parent institution policy related to OSS adoption. Only one library's parent institution and only five libraries have a formal written policy related to adoption of OSS. 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 no sustainability strategy (50 respondents, 70%) or exit strategy (53 respondents, 75%). Strategies included minimizing customizations, providing sufficient staffing with needed expertise, and only adopting systems with good documentation and an active community. Respondents reporting an exit strategy frequently emphasized the criticality of data migration (more than half of relevant comments, 8 of 15).

Survey respondents were asked to identify the open source 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 adopted, the most common being Drupal (6 respondents), Blacklight (5 respondents), Omeka (5 respondents), and DSpace (4 respondents). 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 to 10 with a mean of 2.1 and a median of 2. The number of hours required to support this system ranged from 0 to 512 per month, with a mean of 68 hours and a median of 20 hours.

Only ten (14%) of the respondents were able to track the cost of their most recently adopted OSS system. Of those who could track their costs, expenses ranged from $400 to over $600,000 and, in some cases, represented a multiple year investment. These funds covered a variety of expenses including staff time, hosting, travel, and consulting. The nearly universal primary source of funding was the library's operating budget (69 respondents, 99%).

Respondents were asked 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 included 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 that could provide support for the OSS system (40 responses). Other commonly cited challenges included poor documentation (19 responses), a need for additional training or expertise (16 responses), and substandard development practices (12 responses).

2.1.4 Development

Fifty-six respondents (78%) have contributed to an open source project, including DSpace (12 respondents), Fedora (11 respondents), Hydra (9 respondents), Kuali (6 respondents), Blacklight (5 respondents) and ArchivesSpace (4 respondents). Respondents were asked to describe their contributions to open source projects. Below are some of the highlights.

- 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 (65 respondents), digital preservation (61 respondents), digital asset management (37 respondents), discovery layer (21 respondents), publishing (18 respondents), ILS (18 respondents), and streaming media (16 respondents).
- Where code was contributed, the most common types of projects included institutional repository (32 respondents), digital preservation (22 respondents), digital asset management (20 respondents) and discovery layer (11 respondents).
- Where funding was contributed, the most common projects included institutional repository (18 respondents), digital preservation (19 respondents), and digital asset management (8 respondents).
- Where hosting was contributed, the most common project was digital preservation (9 respondents).

Fifty-six respondents (78%) have contributed to a library related open source project. Of these, respondents were involved in an average of 4.6 projects (median of 3, minimum of 1, maximum of 20), and primary contributors on an average of 1.9 projects (median of 1, minimum of 0, maximum of 20).

Thirty-two respondents identified themselves as the original developer of an open source project. 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.

Eight-one percent of open source contributors (43 respondents) said they were not able to track the costs of their most recent OSS project.

Of the respondents able to identify the source of their open source funding, 96% (43 respondents) said that funds came from their library operating budget. Ten respondents (22%) secured grant money to cover their open source contributions.

Survey respondents were asked to describe the OSS policies used by their library and parent institution. Forty-four (60%) respondents indicated their library has no policy in place for contribution to open source projects, while 20 respondents (27%) have an informal policy. Thirty-four respondents stated that they have no tech transfer policy, while 33 respondents (32%) 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 responses). Other common themes included control of product features and direction (25 responses), and recognition/reputation (14 responses). The most common challenge was allocating sufficient staff time to make meaningful contributions (24 responses). Other commonly cited challenges included writing generalized software for use by a larger community (7 responses) and securing the financial resources needed to support the open source project and community (7 responses).

Since open source project members are rarely collocated, a variety of tools were employed to help coordinate development efforts. Common tools used included shared version control (37 respondents), an issue tracker (36 respondents), a mailing list, (32 respondents), and a wiki (25 respondents). 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, 93%).

The most common licenses used by respondents were GPL v3 (16 respondents), Apache (15 respondents), and Creative Commons (15 respondents).

Respondents were asked to rank a set of success indicators in terms of their importance for the respondent's institution. A significant number (41 respondents, 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 following: staff time commitment required to support the community (41 respondents, 77%); readiness of code quality for public adoption (39 respondents, 74%); and dependence on other internal systems (30 respondents, 57%).

2.1.5 Conclusion

This survey reveals that nearly all responding ARL 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 LIT organizations committed more resources to OSS projects than smaller LIT organizations, but we found no significant correlations suggesting a disproportionate level of commitment to OSS projects as a function of

LIT 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 LIT organizations. Simply stated, LIT organizations that develop software also predominantly contribute to OSS projects.

The results of this survey suggest that we view organizational behaviors surrounding the adoption of open source software separate from contribution to OSS projects. For example, while OSS adoption is viewed by respondents as a means of saving time and resources, OSS contribution is not similarly viewed. Rather, contribution to OSS projects is viewed as being advantageous for different reasons, namely 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 LIT organization.

Control of software emerged as a theme common to both adoption and contribution. Those adopting OSS products 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, LIT organizations perceived a sufficient benefit to their overall productivity to justify the expense of their involvement (as adopters, contributors, or both) in OSS systems.

2.2 Survey Questions and Responses

The SPEC Survey on Open Source Software was designed by Curtis Thacker, Discovery Systems Manager at Brigham Young University's Harold B. Lee Library, Dr. Charles 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.

2.2.1 Survey Response

79 of the 129 ARL Libraries Responded

Total Response Rate - 61%

76 of the 122 Academic – 62% of ARL Academic Institutions

3 of the 6 Governmental - 50% of ARL Governmental Institutions

1 of the 2 Public – 50% of ARL Public Libraries

2.2.2 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

   Number of Library IT staff

Minimum	Maximum	Mean	Median	Std Dev
2	350	25.98	15.0	51.34

Table 2.1: Number of library IT staff.

Minimum	Maximum	Mean	Median	Std Dev
2	50	15.88	14.0	10.17

Table 2.2: Number of library IT staff, academic libraries only.

Minimum	Maximum	Mean	Median	Std Dev
130	350	243.33	250	110.15

Table 2.3: Number of library IT staff, government libraries only.

Minimum	Maximum	Mean	Median	Std Dev
30	30	30	30	N/A

Table 2.4: Number of library IT staff, public libraries only.

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

Software Development Practice	N	Percent
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%

Table 2.5: Software Development Practices ARL libraries participate in.

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

Table 2.6: Methods used by ARL Libraries to provide feedback to library IT staff.

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

2.2.3 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

Reason Cited for Not Releasing System as Open Source	**N**	**Percent**
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%

Table 2.7: Reasons cited for not releasing one or more library specific system as open source.

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

1. Highly customized to address local requirements.
2. Lack of clarity about campus policies for licensing and intellectual property ownership.
3. Legal considerations.
4. Narrow niche applications where a community is unlikely to develop.
5. Not approved for release.
6. Not documented for external audiences.
7. Often these systems reflect local practices. We’ve not viewed them as useful beyond our local environment.
8. Planning to release a service as open source, working on appropriate licensing language at this time.
9. Security

10. Security concerns related to embedded information.
11. Technology Commercialization Office needs to review any software developed at Ohio State University.
12. Time needed for review of and compliance with licenses of third-party components.

2.2.4 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=17

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

2.2.5 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

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**
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

Table 2.8: Implementation/adoption of library specific software.

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 ArchiveSpace 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 Ohio State'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. Please contact us for more information as needed.
- 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 front end 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

Table 2.9: The importance of a given set of criteria used when selecting software.

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 of Congress)
- 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 cyberinfrastructure, 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%

2.2.6 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

Table 2.10: Policies related to OSS.

Comments N=9

- http://uctas.ucop.edu/documents/uc-guidelines-contributing-oss-communities.pdf http://www.ucop.edu/ott/genresources/genguidance.html
- I am not aware of any official or documented policy regarding OSS at the institution at this time.
- LC has policies and procedures for making LC-produced open source code available outside LC. 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-case 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

Table 2.11: Library sustainability and exit strategies.

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

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

2.2.7 Reasons for Adopting OSS

11. Please identify the open source software that has been **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 discovery layer, Fedora Commons Repository, DSpace, Handles, WordPress
- Blacklight, Fedora,
- Blacklight, Hydra, Solr, Fedora Commons, DSpace, Opencast Matterhorn, Avalon Media System, Variations Digital Music Library System. 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
- DSpace, Open Journal System (OJS)
- DSpace
- DSpace, Drupal
- DSpace, and several others
- DSpace, Fedora Commons, Hippo CMS, Drupal, Open Journal Systems
- DSpace, Fedora Commons, Hydra, Apache, MySQL, Solr, Linux, Open Journal System (OJS), Python, R, Ruby, Archivists' Toolkit, ArchiveSpace, WordPress, Drupal, Tomcat
- DSpace, Islandora, Fedora Commons, Drupal, Tesseract, ICA-AtoM, Open Journal System (OJS), Open Book Systems (OBS), Manitobia, LOCKSS, PostgreSQL, MySQL, Apache suite of applications, Python, Redmine, Git

- DSpace, Omeka, MDID
- DSpace, Umlaut, WordPress
- DSpace, Open Journal System (OJS), and VuFind
- DSpace, Open Journal Systems (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 Systems (OJS), Open Conference Systems (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 Systems (OJS), Open Conference Systems (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 web Server, 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, Archivists' ToolKit, VireoCat, various open source utilities
- Hydra, Fedora Commons, Solr, Blacklight, phpScheduleIt, Open Harvester, WordPress, others.
- 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 Systems (OJS) and Omeka; CORAL
- Open Journal Systems (OJS), DSpace, Omeka
- Open Journal Systems (OJS), Open Monograph Press (OMP), Drupal, WordPress, Dokuwiki, MediaWiki, Islandora, Fedora Commons, Spiceworks, PWik, Omeka, Archivists' Toolkit
- Omeka, Avalon media System, 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 Systems (OJS)
- phpScheduleIT, Omeka, WordPress, Archon, ArchivesSpace, Blacklight, SubjectsPlus, Variations Digital Music Library System, Avalon Media Server, 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
- Hydra, Omeka, Drupal, Shibboleth
- Public Knowledge Project (PKP), Research Project Calculator (Assignment Calculator), ArchivesSpace, Apache, Linux, MySQL, PostgreSQL, Hydra, Blacklight, Fedora, Solr, PersistantURLs (PURLZ), Omeka, Open Journal Systems (OJS)
- Streetprint, DSpace, OS Ticket, DokuWiki, Guide on the Side
- DuraSpace products, SugarCRM, ArchiveSpace
- The main library-specific OSS we use: VuFind, 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 Systems (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, Public Knowledge Project (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

Table 2.12: Reasons for adopting OSS over a competing vended product.

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 Moderately 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=5

- As a federal agency LC must be very cautious about appearing to endorse one type of product over another, hence has not provided answers to question no. 8.
- NOTE: 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=64

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 ca. 25 hr/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.

OSS System	Staff	Staff hours	Comments
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	
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

OSS System	Staff	Staff hours	Comments
			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/cuss for new website from scratch.
I don't have the details			
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.
Islandora	4	160	We have four full time staff developing on the Islandora stack. This includes efforts for Drupal, Solr, and Fedora, which comprise Islandora.
Koha	7	130	
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.

OSS System	Staff	Staff hours	Comments
obento (our in-house developed bento search)	4	500 (approx.)	
Open Journal System (OJS)	3	100	
Omeka	1.5	40	
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 Systems (OJS)	2	400	
Papyrus	2	210	
ResourceSpace	1	8	
Room Booking	2	60	
RUanalytic	3	400	
phpScheduleIT	4	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.

OSS System	Staff	Staff hours	Comments
Vireo	2	200	Times are grossly estimated for the last question.
Wireo	2	120	
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	

Table 2.13: Number of staff and staff hours to adopt an OSS projects.

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=58

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 2 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.

OSS System	Staff	Staff hours per month	Comments
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	Really strange question, especially related to the previous question.
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	
I don't have the details			
ICA - AtoM	2	20	

OSS System	Staff	Staff hours per month	Comments
Islandora	1	70	
Islandora	2		See above comment.
Nagios	0.25	1	For this OSS component, there only requires minimal effort to maintain, just the application of system patches.
			Not sure how this differs from above. The distinction between these two is unclear to us.
obento (our in-house developed bento search)	2	20	
Open Journal System (OJS)	1	10	
Open Journal Systems (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 Systems (OJS)	1	8	Hours/Staff do not include continued development time.
Open Journal Systems (OJS)	2	44	

OSS System	Staff	Staff hours per month	Comments
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.
same			
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.
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	
Vireo	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

OSS System	Staff	Staff hours per month	Comments
			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.

Table 2.14: Number of staff and staff hours required to maintain an OSS project.

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.

2.2.8 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.
$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

Cost	**Expenses Covered**
$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

Table 2.15: Reported costs of adopting an OSS system.

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%

Table 2.16: Reported sources of funding for OSS systems.

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.
- Note: 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.

2.2.9 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

Benefit 1	**Benefit 2**	**Benefit 3**
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	In line 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 w/ 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.

Benefit 1	**Benefit 2**	**Benefit 3**
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

Benefit 1	**Benefit 2**	**Benefit 3**
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 CSU's data than we do with vended products.	OSS communities tend to have vibrant and engaged members, which can be a good support resource.

Benefit 1	**Benefit 2**	**Benefit 3**
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 contributed 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 systems	Research and publishing opportunities

Benefit 1	**Benefit 2**	**Benefit 3**
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

Table 2.17: Reported benefits of adopting OSS.

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

Challenge 1	**Challenge 2**	**Challenge 3**
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

Challenge 1	**Challenge 2**	**Challenge 3**
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	Incomplete functionality - Develop alternative workflows, contribute enhancements	Poorly developed or managed code contribution process - Minimize customization of software

Challenge 1	Challenge 2	Challenge 3
reports, and documentation to project		
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	Alignment of local project timelines with those of OSS products

Challenge 1	**Challenge 2**	**Challenge 3**
	documentation and knowledge bases	
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 a OSS software.
Keeping up with software updates	Training overhead for new staff	
Lack of documentation - communication on listserves 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.

Challenge 1	**Challenge 2**	**Challenge 3**
Maintain thorough documentation of local implementation & customization decisions	Failsafe upgrades: need to make sure locally developed plugins, etc. don't crash w/ 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 which 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.		

Challenge 1	**Challenge 2**	**Challenge 3**
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 products	Product looks behind-the-times
Staff Cost	Long term stability and robustness of software	Open source licenses can be variable

Challenge 1	**Challenge 2**	**Challenge 3**
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 1 or 2 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.

Challenge 1	Challenge 2	Challenge 3
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.		

Table 2.18: Reported challenges of adopting OSS.

2.2.10 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 Digital Music Library System (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
- DSpace
- DSpace
- DSpace and File Analyzer
- DSpace, Kuali, Fedora, 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
- Fedora Commons
- Fedora Commons, DuraSpace, ArchivesSpace
- Fedora Commons, Blacklight, Hydra, Avalon Media System, Hydramata, ArchiveSpace, APTrust, DPN, SOLR-Marc, Tracksys.

- Fedora, Islandora
- Guide on the side
- Hydra
- Hydra
- 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
- Hydra, Blacklight

- SRA toolkit, BLAST, C++ toolkit, variety of scientific tools
- SubjectsPlus, Remixing Archival Metatdata Project (RAMP); Variations Digital Music Library System, Avalon Media System, 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
Blogging	2	2	1	—	40	44

Type of OSS Project	Code (i.e., developer time)	Money	Hosting	Other contribution	N/A	N
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

Table 2.19: Ways libraries reported they are contributing to OSS.

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 (see question #8 for more detail on Omeka_a11y), 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 listserves.
- 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. Variations Digital Music Library System, Avalon Media System - 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 & testing (Kaltura). Digital preservation: we manage & offer fee-based support 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/listserves
- 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 Temple University Libraries is 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 which 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

Table 2.20: OSS projects libraries have contributed to and initiated.

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

Number of Library Staff	Percentage 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

Number of Library Staff	Percentage of Time
2	10
2	20
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

Number of Library Staff	Percentage of Time
10	50
10	60
12	varies
14	50

Table 2.21: The number of library staff and about what percent of their time are dedicated to contributing to the development of OSS projects.

	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

Table 2.22: Distribution of the number of library staff and about what percent of their time are dedicated to contributing to the development of OSS projects.

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 System, 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 scholar 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 Univ 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, Islandora
- 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 repo.

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

Reasons	**1 Not Important**	**2**	**3**	**4**	**5 Very Important**	**N**
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

Table 2.23: The importance of a common set of reasons used to decide to open source a project.

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

2.2.11 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 Systems (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

Table 2.24: Reported costs of contributions made by ARL libraries to OSS projects.

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

Funding Source	**N**	**Percent**
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%

Table 2.25: Reported funding sources for OSS contributions.

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

- Funded by another university division (Technology Services)
- NOTE: Able to track, chose not to track. Would come from library’s operating budget.

2.2.12 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

Benefit 1	**Benefit 2**	**Benefit 3**
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	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.

Benefit 1	Benefit 2	Benefit 3
always be the technical possibility.		
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 which 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		

Benefit 1	**Benefit 2**	**Benefit 3**
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 U.S. 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	

Benefit 1	**Benefit 2**	**Benefit 3**
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.	

Table 2.26: Reported benefits of contributing to OSS projects.

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

Challenge 1	**Challenge 2**	**Challenge 3**
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	

Challenge 1	**Challenge 2**	**Challenge 3**
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	

Challenge 1	**Challenge 2**	**Challenge 3**
	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% there 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

Challenge 1	Challenge 2	Challenge 3
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 but 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.		

Table 2.27: Reported challenges of contributing to OSS projects.

2.2.13 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

Repository	N
GitHub	38
Google Code	3

SourceForge	3
Bitbucket	2
Drupal GIT	1
RedMine	1
Subversion	1

Table 2.28: Code repository or forge used by responding libraries.

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

Collaboration Tool	**N**	**Percent**
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%

Table 2.29: Collaboration tools used by respondents.

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

2.2.14 Licensing Model for Distribution of OSS

29. What licensing models does your organization recommend for distribution of software?

    Check all that apply. N=42

OSS License	**N**	**Percent**
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%

Table 2.30: OSS licenses used by respondents.

Please briefly describe the other licensing model. N=12

- Educational Community License (ECL) - ECL 2

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

2.2.15 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

Reasons	**1 Not Important**	**2**	**3**	**4**	**5 Very Important**	**N**
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

Table 2.31: Reported indicators that a contribution to an OSS project has been successful.

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.
- Did not really understand the question.
- LC did not reply to question no. 18 because as a federal agency we are very cautious about appearing to favor one kind of product, e.g., OSS, over another, e.g., vended software.

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

2.2.16 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 5 years, and the OSS products have not historically had the functions we required. That may be changing looking forward.

2.2.17 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.
    - LC did not respond to Question 10 because we are very cautious about replying to questions that involve any comparison among products or types of products, since they could become objects for federal contracting.

- OSS allows for greater customizations that fulfill the needs of so many UCI 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 UC 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 CSU Libraries and Academic Computing and Networking Services (ACNS) both report to D. Patrick Burns, 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 devoted several full time staff to developing Fedora 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 center, 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.

Chapter 3

Barriers to Initiation of Open Source Software Projects in Libraries[2]

3.1 Abstract

Libraries share a number of core values with the Open Source Software (OSS) movement, suggesting there should be a natural tendency toward library participation in OSS projects. However, Dale Askey's 2008 Code4Lib column entitled *We Love Open Source Software. No, You Can't Have Our Code*,[3] claims that while libraries are strong proponents of OSS, they are unlikely to actually contribute to OSS projects. He identifies, but does not empirically substantiate, six barriers that he believes contribute to this apparent inconsistency. In this study we empirically investigate not only Askey's central claim but also the six barriers he proposes. In contrast to Askey's assertion, we find that initiation of and contribution to OSS projects are, in fact, common practices in libraries. However, we also find that these practices are far from ubiquitous; as Askey suggests, many libraries do have opportunities to initiate OSS projects, but choose not to do so. Further, we find support for only four of Askey's six OSS barriers. Thus, our results confirm many, but not all, of Askey's assertions.

3.2 Motivation

The mission statement of the American Library Association includes the charge to "ensure access to information for all."[4] This charge comes without restriction, cost or qualification. Stated another way, libraries make information freely available to all, regardless of how that information is to be used. Similarly, open source software (OSS) "licenses must permit non-exclusive commercial exploitation of the licensed work, must make available the work's source

[2] This chapter is published in the code4lib journal, 2015 [46].
[3] See http://journal.code4lib.org/articles/527
[4] See http://www.ala.org/aboutala/missionpriorities

code, and must permit the creation of derivative works from the work itself" [47]. The core values of libraries and the OSS movement are similar, suggesting that libraries should tend to favor the OSS model. In particular, they might feel a responsibility to share the code they have developed with other libraries in a spirit of openness and access for all.

That libraries are predisposed to OSS adoption and contribution is not a new idea. Pat Eyler, an open source developer for the Koha ILS project, said "That more librarians aren't actively using and evangelizing free software is an indictment against us for not letting them in on our secret" [41]. Nicole Engard characterized the issue this way: "It has been suggested that libraries are almost ethically required to use, develop and support open source software" [43].

Richard Stallman, the pioneering free software evangelist, stated that "… universities shouldn't be developing proprietary software. It is better if they develop none at all, because [by doing so] they are betraying their mission to contribute to human knowledge" [42].

Despite the suggestion that libraries are ethically obligated to use and create OSS, it has been observed that libraries seem reluctant to share their code. In 2008 Dale Askey authored a column in this journal entitled *We Love Open Source Software. No, You Can't Have Our Code*. He states that "Librarians are among the strongest proponents of open source software. Paradoxically, libraries are also among the least likely to actively contribute their code to open source projects" [44]. Askey identified a list of six issues he believes contribute to this dichotomy. In his own words:

After pondering this issue for some time, I identified the following issues as the driving forces that undermine the sharing of open source software in libraries:

- *perfectionism – unless the code is perfect, we don't want anyone to see it*
- *dependency – if we share this with you, you will never leave us alone*

- *quirkiness – we'd gladly share, but we can't since we're so weird*
- *redundancy – we think your project is neat, but we can do better*
- *competitiveness – we want to be the acknowledged leader*
- *misunderstanding – a fundamental inability to understand how an open source community works*

Many of these issues operate in combination, but any one of them is sufficient to thwart the development and adoption of open source software in libraries.

In this paper, we report on our empirical investigation into Askey's central claim. We examine the six barriers he proposes in light of our empirical results.

3.3 Methods

The Association of Research Libraries (ARL) "is a nonprofit membership organization of 125 research libraries in North America. The Association operates as a forum for the exchange of ideas and as an agent for collective action." Each year ARL distributes and publishes a small number of surveys, called SPEC Kits, that are proposed and designed by librarians and other interested parties.

In February 2014, ARL distributed a 32-question survey authored by Curtis Thacker, Charles Knutson, and Mark Dehmlow, to 127 member libraries. Seventy-seven libraries (61%) responded to the survey, the results of which were subsequently published as *SPEC Kit 340: Open Source Software [45]* (hereafter referred to as "the SPEC survey").

The purpose of the SPEC survey was to study ARL member libraries' adoption and/or development of OSS for the primary functions carried out in libraries. We aimed to understand organizational factors that affect decisions to adopt OSS. With regard to development of OSS, we studied: 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.

Questions were reviewed, evaluated and refined by empirical software engineering researchers from the SEQuOIA[5] Lab in the Brigham Young University Computer Science Department. This exercise enabled us to deepen our understanding of issues related to open source software development by applying the growing body of work in the area of empirical software engineering. The creation of the survey instrument followed best practices for empirical software engineering surveys [48].

Questions were crafted to empirically test several of the issues laid out in Askey's column. In particular, the following question provided respondents with an opportunity to identify reasons for not openly releasing software they had developed:

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

Yes
No

If yes, what are the primary reason for not releasing it as open source? Check all the apply.

- Concerns about staff time commitment required to support the community
- Concerns that the code quality is not ready for public adoption
- Dependence on other internal systems
- It didn't occur to us

[5] SEQuOIA = "Software Engineering Quality: Observation, Insight, Analysis"

- Seeking to license or sell the system
- A competitive desire to have the best system
- Other reasons

Figure 3.1: A sample question for the SPEC Survey.

The table below illustrates the relationship between the options presented in the question and the issues presented by Askey. The first column identifies each issue as presented in the survey, while the second column presents the issues as stated by Askey. Two of the issues offered by Askey were not tested because they fell outside the scope of the SPEC survey. Two other issues were added in an attempt to validate additional reasons for which an institution might choose not to open source their code. Of these two issues, the second one ("seeking to license or sell the system") was inspired by a response[6] made to Askey's column.

Barriers to Initiation of Open Source Software Projects in Libraries	**Barriers to Initiation of Open Source Software Projects in Libraries**
Concerns that the code quality is not ready for public adoption	perfectionism – unless the code is perfect, we don't want anyone to see it
Concerns about staff time commitment required to support the community	dependency – if we share this with you, you will never leave us alone
Dependence on other internal systems	quirkiness – we'd gladly share, but we can't since we're so weird
This issue was not addressed in the survey since it deals more with the adoption of OSS rather than contribution to or initiation of an OSS project.	redundancy – we think your project is neat, but we can do better
A competitive desire to have the best system	competitiveness – we want to be the acknowledged leader
This issue is a catch-all and was addressed by	misunderstanding – a fundamental inability to

[6] See http://journal.code4lib.org/articles/527#comment-1299

the other options presented in this question, as well through other questions presented the SPEC survey.	understand how an open source community works
Seeking to license or sell the system	N/A
It didn't occur to us	quirkiness – we'd gladly share, but we can't since we're so weird

Table 3.1: A mapping between Askey's claims and the issue as stated in the SPEC Survey.

ARL reviewed and administered the survey. Participants were given four weeks to respond and ARL sent two email reminders as the deadline approached. A spreadsheet of the complete response data was returned to the authors for analysis and preparation for publication.

Survey results were reviewed and statistically analyzed. Free response questions were encoded and qualitatively analyzed for themes and best practices. The executive summary of the SPEC survey includes an overview of statistical results that spans the entire survey. A specific set of results relevant to this paper are presented and discussed in the sections below.

3.4 OSS Adoption

Askey's initial premise is that libraries love OSS. He cites Dan Chudnov [49] who asserts that infrastructure software and programming languages are widely adopted by libraries. Operating systems such as Linux, web servers such as Apache, and programming languages such a Ruby and Java are examples of OSS systems commonly adopted by libraries. These applications compete with commercial applications for market share and often hold the largest slice of the pie. Askey also pointed out that OSS adoption is ubiquitous for other common types of software applications such as web browsers (such as Mozilla) and mail clients (such as Thunderbird). Market share statistics for Linux[7], Apache[8] and Mozilla[9] substantiate these claims.

[7] See operating system statistics at http://www.netmarketshare.com/

[8] See http://news.netcraft.com/archives/2015/01/15/january-2015-web-server-survey.html

[9] See http://gs.statcounter.com/#all-browser-ww-monthly-201502-201502-bar

The SPEC survey found that 74 respondents (97%) had deployed open source software in their libraries, suggesting that, at least for ARL Libraries, adoption of OSS is essentially ubiquitous. This data strongly supports Askey's claim that libraries love OSS. We also wanted to understand the specific types of OSS that are loved by libraries.

Askey asserts that libraries have "strongly embraced...object repositories such as DSpace and Fedora and content management systems such as Drupal." SPEC survey respondents were invited to provide information about the type of software being used for various purposes. Respondents most frequently reported choosing OSS solutions for institutional repositories (52 total), blogging (51 total) and digital preservation (50 total). See the table below for more details on how respondents have adopted OSS within their institutions.

Purpose of System	**Respondents using a system for this purpose**	**Respondents using an OSS solution**	**Percent respondents using an OSS solution**
Institutional repository	69	52	75.4%
Blogging	65	51	78.5%
Digital preservation	57	50	87.7%
Publishing	57	42	73.7%
Authentication/identity management	67	35	52.2%
Digital asset management	64	34	53.1%
Web analytics	71	22	31.0%
Discovery layer	73	19	26.0%
Study room scheduler	59	18	30.5%
ELMS	23	17	73.9%
Streaming media	62	17	27.4%
Data warehouse	26	14	53.8%

Visualization	27	10	37.0%
Electronic resource management	71	9	12.7%
Link resolver	70	9	12.9%
Floor maps	43	8	18.6%
Data analysis	32	7	21.9%
ILS	74	6	8.1%
Course reserve	68	4	5.9%
Inter-library loan	73	3	4.1%

Table 3.2: Adoption of various types of library OSS.

The SPEC survey confirmed Askey's sense that DSpace and Fedora were "strongly embraced" by libraries. Sixty-six respondents reported the OSS projects they had adopted. We found that the most commonly adopted open source systems were DSpace (31 respondents, 47%[10]), Fedora (21 respondents, 32%), Open Journal System (19 respondents, 29%), Blacklight (14 respondents, 21%), Hydra (12 respondents, 18%), Vufind (8 respondents, 12%), ArchivesSpace (7 respondents, 11%) and Archivist Toolkit (6 respondents, 9%).

The SPEC survey revealed compelling evidence for the widespread adoption of library specific software, even beyond Askey's claims.

Respondents were further asked to describe three benefits and three challenges associated with adopting OSS. The most commonly reported benefit was the ability to customize the software (50 responses). Other common themes included low cost or time to implement (27 responses) and association with an active community (27 responses). The most common challenge was the need for highly skilled staff that could provide support for the OSS system (40 responses). Other commonly cited challenges included poor documentation (19 responses), a

[10] Percentages are based on the 66 respondents who reported the OSS projects they had adopted.

need for additional training or expertise (16 responses), and substandard development practices (12 responses).

3.5 OSS Contribution

Askey shares his perception that libraries are reluctant to initiate and/or contribute to OSS projects, despite their nearly universal enthusiasm for adoption. Askey's main claim is: "where we tend to fall flat is in the area of creating, maintaining, and sharing library-specific applications. There are certainly myriad exceptions to this statement, but I would suggest that however large and noteworthy, they remain the exceptions, and not the rule" [44]. While Askey's statement mainly addresses initiation of OSS projects, maintaining library-specific applications could be interpreted as contribution to OSS projects.

Askey's column focused primarily on contributions to OSS projects in the form of source code. Beyond software, OSS projects benefit from many types of contributions including, money, hosting, testing, etc. The table below shows the types of contributions that libraries have made to OSS projects.[11]

Type of OSS Project	**Code (i.e., developer time)**	**Money**	**Hosting**	**Other contribution (e.g., testing, requirements)**
Institutional repository	32 (57%)	18 (32%)	5 (9%)	10 (18%)
Digital preservation	22 (39%)	19 (34%)	9 (16%)	11 (20%)
Digital asset management	20 (36%)	8 (14%)	4 (7%)	5 (9%)
Discovery layer	11 (20%)	3 (5%)	2 (4%)	5 (9%)
Publishing	5 (9%)	5 (9%)	5 (9%)	3 (5%)
ILS	6 (11%)	5 (9%)	—	7 (13%)
Streaming media	7 (13%)	4 (7%)	2 (4%)	3 (5%)

[11] All percentages are based on the 56 respondents who have contributed to one or more OSS project. All 56 of these respondents reported on the types of OSS contributions they made.

Study room scheduler	5 (9%)	—	—	1 (2%)
Link resolver	3 (5%)	1 (2%)	1 (2%)	1 (2%)
Authentication/ identity management	8 (14%)	—	1 (2%)	2 (4%)
Inter-library loan	2 (4%)	1 (2%)	3 (5%)	3 (5%)
Data analysis	5 (9%)	1 (2%)	2 (4%)	2 (4%)
Blogging	2 (4%)	2 (4%)	1 (2%)	—
Electronic resource management	6 (11%)	—	2 (4%)	4 (7%)
Course reserve	4 (7%)	—	—	2 (4%)
Floor maps	4 (7%)	—	1 (2%)	1 (2%)
Data warehouse	6 (11%)	—	2 (4%)	1 (2%)
ELMS	3 (5%)	1 (2%)	—	1 (2%)
Visualization	4 (7%)	1 (2%)	1 (2%)	2 (4%)
Web analytics	3 (5%)	—	1 (2%)	1 (2%)
Other type of project	15 (27%)	5 (9%)	2 (4%)	6 (11%)

Table 3.3: Reported Contributions to OSS projects

The SPEC survey found that 56 respondents (78%) had contributed to one or more open source projects; of these, 50 respondents indicated which projects they had contributed to. The most common projects included DSpace (12 respondents, 24%[12]), Fedora (11 respondents, 22%), Hydra (9 respondents, 18%), Kuali (6 respondents, 12%), Blacklight (5 respondents, 10%) and ArchivesSpace (4 respondents, 8%). The SPEC survey found that respondents had contributed to an average of 2.6 OSS projects and a median of 1 OSS project. These findings support Askey's claim that contribution to OSS by libraries is common, yet far from universal.

3.6 OSS Initiation

[12] Percentages are based on the 50 respondents who indicated which projects they had contributed to.

Askey addressed initiation of OSS when he claimed that “where we tend to fall flat is in the area of creating, maintaining, and sharing library-specific applications. There are certainly myriad exceptions to this statement, but I would suggest that however large and noteworthy, they remain the exceptions, and not the rule” [44].

Thirty-two (42%) respondents identified themselves as the original developer of an open source project. Respondents initiated an average of 1.4 OSS projects and a median of zero OSS projects. Thus we see that while a number of institutions have some experience initiating OSS projects, initiation is far from the norm. Our finding supports Askey’s claim.

Respondents were asked if any of their in-house software could have been, but had not yet been, released under an open source license. Fifty-three respondents (69%) answered in the affirmative. Additionally, the SPEC survey revealed libraries that always choose to share their sharable projects, and, conversely, there are libraries that could share but have thus far not chosen to share their code. The table below breaks down these responses in greater detail.

Position on OSS Project Initiation	**Number of Respondents**	**Percent of respondents**
Nothing to share	18	23%
Could but didn’t	52	68%
Sometimes share	24	31%
Never share	28	36%
Always share	7	10%
Total respondents	77	100%

Table 3.4: The initiation practices of responding libraries.

Respondents cited all of Askey’s barriers as reasons for not open sourcing a sharable system. We address each of these issues in the sections below.

3.6.1 Perfectionism

Thirty-nine (74%) of those who chose not to open source their code cited “concerns that the code quality is not ready for public adoption.” The perception that the code quality is not acceptable, and therefore cannot be shared, is very common.

This particular question in the SPEC survey was only able to test perceptions of libraries. As pointed out by Askey, intrinsic to the open source philosophy is the idea that the community will improve upon an initial system. Linus' Law, as described by Raymond [50], describes OSS communities this way: "given enough eyeballs, all bugs are shallow", or more formally: "Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix will be obvious to someone." It follows from Linus’ Law that not sharing code due to quality issues is more a matter of pride than practicality.

3.6.2 Dependency

“Nothing is more certain in the world than this: if you share software with someone, you will be asked to support it, even if you make it perfectly clear that you have no ability and no intention to do so” [44]. Forty-one respondents (77%) cited “staff time commitment required to support the community” as a reason for not open sourcing a product that could have otherwise been shared. The SPEC survey offers strong evidence that the perception of dependency is a common barrier among ARL members.

3.6.3 Quirkiness

Quirkiness is defined by Askey as “the sense that one organization’s needs are so locally-tailored that [it] would make no sense to release the software to the broader library community.” Later in the same section he cites an example of quirkiness as dependence on “idiosyncratic local metadata scheme.” The SPEC survey addresses quirkiness in three ways. First, 30 respondents (57%) cited “dependence on internal systems” as a reason for not open sourcing a system that could have otherwise been open sourced. Second, 7 respondents (13%) stated “it didn’t occur to us” as a reason for not open sourcing their software. Third, the issue of quirkiness was directly addressed by respondents who entered free form responses describing reasons they chose not to open source a system. Responses included: “Highly customized to address local requirements”; “Narrow niche applications where a community is unlikely to develop”; and “Often these systems reflect local practices. We’ve not viewed them as useful beyond our local environment.” These data are evidence of quirkiness among ARL members and support Askey’s claims.

3.6.4 Redundancy

Redundancy, as described by Askey, “is when there is perfectly acceptable software available and yet is rejected because it’s not quite what one would have done had they created the software.” We found that this issue relates more to adoption than initiation of OSS. As a consequence, we did not study this issue in detail.

3.6.5 Competitiveness

Askey explains that libraries tend to implement their own systems (e.g., institutional repository, digital libraries, and web services) because they “want to be the acknowledged leader.” While one respondent of the SPEC survey indicated “a competitive desire to have the best system” as a reason for not open sourcing their software, no other respondent cited such motivation. As a result, while we find some support for Askey’s claim, competitiveness does not appear to be widespread.

3.6.6 Misunderstanding

Askey describes misunderstanding as “a fundamental inability to understand how an open source community works.” We determined that “misunderstanding” primarily suggested that respondents did not understand the benefits of involvement with an OSS community. This issue represents a catch-all of sorts that encompasses the other issues we've discussed. The breadth of "misunderstanding" prevented us from testing this issue in the same manner as the other issues presented above. Other questions in the survey do, however, offer insights into the benefits libraries currently enjoy as a result of adoption of and contribution to library-specific OSS projects. We highlight some of these insights below.

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

Control of software emerged as a theme common to both adoption and contribution. Those adopting OSS products 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 libraries contributing to OSS projects felt that they gained greater opportunity to influence product direction, especially with respect to software features. In both cases, library information technology organizations perceived a sufficient benefit to their overall productivity to justify the expense of their involvement (as adopters, contributors, or both) in OSS systems.

When asked about reasons for open sourcing their project, SPEC survey 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). The experiences shared by respondents who initiated an open source system support the idea that one way to inject quality into a system is to open source it. In contrast to Askey's claim, there were many respondents who demonstrated an understanding of this benefit of open sourcing their code. Additionally, of the 54 respondents who have a system they chose not to release as open source, 24 (44%) have initiated at least one open source project. Further research is required to understand the motivation of these ARLs decision to share one system but not another.

Many respondents expressed a desire on the part of their developers to share with and participate in one or more OSS communities. Larger LIT organizations committed more resources to OSS projects than smaller LIT organizations, but we found no significant correlations suggesting a disproportionate level of commitment to OSS projects as a function of LIT 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 LIT organizations. Simply stated, LIT organizations that develop software have also generally contributed to one or more OSS projects.

3.7 Additional Insights

In the final section of his column Askey makes several suggestions on what should be done to overcome the issues he discusses. We address a few of these suggestions in this section.

In 2008 Askey claimed that there was no standard way of distributing library specific code, suggesting that a single place should be agreed upon as the established method for sharing code. GitHub has emerged as the preferred method for many open source projects (including libraries) to share their code. GitHub accommodates large OSS projects such as Fedora, DSpace, Hydra and others as well as supporting what Askey calls OSS lite[13]. Forty-one SPEC survey respondents indicated that they use a public forge to manage and share their open source projects. Thirty-eight of these use GitHub for this purpose. While making use of an open source forge, such as GitHub, to share code is effective, it is unclear whether this tool has impacted the propensity of libraries to initiate an OSS project.

Askey states that "libraries that wish to use open source software need to understand the staffing commitment they are making by going that route. Open source software requires programmers, interface designers, and system administrators." In our review of organizations that contribute to open source projects, software development staff ranged from one or two to as many as fourteen. 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 that didn't have resources for large technology staffs could still contribute to projects with as few as one or two programmers. The median number of staff reported as working on OSS projects was two, with an average of nearly four.

The results of the SPEC survey suggest that we view organizational behaviors surrounding the adoption of open source software separate from contribution to OSS projects. For example, while OSS adoption is viewed by respondents as a means of saving time and resources, OSS contribution is not similarly viewed. Rather, contribution to OSS projects is viewed as being advantageous for different reasons, namely 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 LIT organization.

[13] Askey defines OSS Lite as "tiny programs written in various scripting languages that drive all the doodads and widgets on our Websites, or extend (or, in some cases, repair) the functionality of our commercial systems."

3.8 Threats to Validity

Care must be taken when generalizing survey findings to a larger population. The SPEC survey was distributed to all 127 ARL member libraries. ARL libraries are often considered a model for best practices, but are not a representative set of research libraries or libraries in general. Further, the 77 respondents of the survey self-selected, introducing bias toward libraries that are interested or invested in OSS. Also, survey fatigue is a large concern. The SPEC survey was relatively long (32 questions), with some questions involving multiple parts and some requiring respondents to look up specific information in order to answer. Several instances were found where respondents didn't answer questions completely, which can be seen in the tables above.

3.9 Future Work

The SPEC survey revealed that there are libraries that always choose to share their sharable projects, and, conversely, there are libraries that could share their code but have never chosen to. Future work could include looking for correlations between a library's software engineering, talent management and innovation policies and practices, and its propensity to initiate OSS projects.

In the years since the publication of Askey's column two significant types of organizations have arisen within the library landscape, exerting considerable influence on open source software projects. Governing foundations, such as DuraSpace, Kuali, the Islandora Foundation, the Software Conservancy Foundation and ArchivesSpace, manage requirements and coordinate resources of member libraries. Supporting vendors, such as Bywaters and @mire, offer support and hosting services to OSS adopters. While outside the scope of the research we performed, the impact of such organizations is highly relevant to the issues posed by Askey and warrants further investigation.

3.10 Conclusion

We found support for many of the issues presented in Askey's column. The majority of SPEC survey respondents have adopted and/or contributed to at least one OSS project. Nearly half of respondents chose to initiate one or more OSS project. While most institutions have some experience with OSS, most have only made an initial foray into the space. As Askey suggests, many libraries do have opportunities to initiate OSS projects, but choose not to do so. We found strong evidence supporting the existence of "perfectionism," "quirkiness," "dependency" and "misunderstanding," however, "competitiveness" was extremely rare. Thus, we find support for many, but not all of Askey's assertions.

The emergence of GitHub as a preferred means of sharing code was highlighted as a development since Askey's 2008 column. We would suggest that library information technology organizations participating in OSS projects typically understand that they must dedicate technical personnel and other resources in order to do so. Finally, we found that OSS comes with a number of financial trade-offs that need to be carefully examined when considering adoption, contribution and initiation of OSS projects.

3.11 Acknowledgements

One of the authors (Curtis Thacker) spoke with Mr. Askey about his column and the work we were doing on this paper at the CNI Spring 2015 Membership Meeting in Seattle. We appreciated his encouragement and insights in addition to his thought-provoking column which contributed inspiration for both the SPEC survey and this paper.

Chapter 4

Toward Understanding the Propensity of Libraries to Initiate Open Source Software Projects

4.1 Abstract

Libraries share a number of core values with the Open Source Software (OSS) movement, suggesting that there should be a natural tendency toward library participation in OSS projects. However, our study suggests that while libraries frequently use and contribute to OSS, they often choose not to initiate OSS projects leveraging code they have created for internal purposes. The goal of this paper is to empirically investigate possible correlations between a library's policies and practices in software engineering, talent management and innovation, and its propensity to initiate open source software projects.

4.2 Introduction

Libraries rely heavily on software to carry out their basic business functions. Much of this software is Commercial off the Shelf (COTS), however adoption of Open Source Software (OSS) has become a viable option. There are many library specific open source software projects. The adoption of, contribution to and initiation of OSS projects in the Library Information Technology (LIT) context is only beginning to be studied.

The mission statement of the American Library Association includes the charge to "ensure access to information for all." This charge comes without cost or qualification. Stated another way, libraries make information freely available to all regardless of how that information will be used. Similarly, 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" [47]. Information sharing and open standards are among the values shared by the OSS movement and Libraries [51]. This confluence of core values suggests that libraries should tend to favor the OSS model. In particular, they may feel a responsibility to share the code they have developed with other libraries in a spirit of openness and access for all.

The predisposition of libraries toward OSS adoption and contribution is not a new idea. Pat Eyler, a developer for the widely-adopted open source integrated library system Koha, said: “That more librarians aren’t actively using and evangelizing free software is an indictment against us for not letting them in on our secret” [41]. Richard Stallman, the pioneering free software evangelist, added that “… universities shouldn’t be developing proprietary software. It is better if they develop none at all, because [by doing so] they are betraying their mission to contribute to human knowledge” [42]. Finally, Nicole Engard characterized the issue this way: “It has been suggested that libraries are almost ethically required to use, develop and support open source software” [43].

Despite the suggestion that libraries are ethically required to use and create OSS, it has been observed that libraries seem reluctant to share their code. In 2008, Dale Askey remarked that: “Librarians are among the strongest proponents of open source software. Paradoxically, libraries are also among the least likely to actively contribute their code to open source projects” [44]. Further, Askey identified a list of six interrelated issues that he believes contribute to this dichotomy. In his own words:

- *perfectionism – unless the code is perfect, we don’t want anyone to see it*
- *dependency – if we share this with you, you will never leave us alone*
- *quirkiness – we’d gladly share, but we can’t since we’re so weird*
- *redundancy – we think your project is neat, but we can do better*

- *competitiveness – we want to be the acknowledged leader*
- *misunderstanding – a fundamental inability to understand how an open source community works*

Thacker et al. authored a targeted survey, consisting of 32 questions aimed at studying ARL member libraries' adoption and/or development of OSS for the primary functions carried out in libraries. They wanted to understand organizational factors that affect decisions to adopt OSS, and test Askey's assertions. With regard to development of OSS, they studied: 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. In February 2014, the Association of Research Libraries (ARL) distributed the survey to 127 member libraries. Seventy-seven libraries (61%) responded, and the results were subsequently published as SPEC Kit 340 [8]. Thacker et al. were able to empirically test Askey's assertions, and offered support for Askey's primary claim, as well as many of the contributing factors he identified [45, 46].

Among Thacker et al.'s findings was that 69% of respondents had developed library specific systems that could, but had not been released as open source. In this paper, we revisit this result, and study policies and practices in the areas of software engineering, talent management, and innovation and R&D looking for correlation that offer insights into a library's motivations regarding initiation of OSS projects. We add to SPEC Kit 340 data collected via two other related SPEC Kits, and use statistical and data mining methods to bring out relevant insights and discuss our findings in light of current software engineering and OSS research, both as they apply in general and in the specific context of libraries.

4.3 Background

DeLone and McLean suggest six interrelated measures of information system success: system quality, information quality, use, user satisfaction, individual impact, and organizational impact [52-54]. Crowston et al. revisit these measures and suggest a set of measures that apply to the OSS process including: movement from alpha to beta to stable, achievement of identified goals, developer satisfaction, number of developers, developer level of activity, time between releases, time to close bugs or implement features, individual job opportunities and salary, individual reputation, and knowledge creation [55].

Much has been written about the motivation of contributors to general OSS projects [7, 10-15, 56-60]. Intrinsic motivations such as learning and altruism tend to be most effective. Career advancement and reputation are also common motivators. Choi, et al. reported that altruism and learning are the top two motivations for Library OSS developers [61]. Other motivations include fun, personal needs, extrinsic rewards and future returns.

West and O'Mahony describe two ways that OSS projects are initiated [32]. Community-driven projects are founded and managed within the context of a community. Spinout projects occur when "a sponsor of an internally developed software project releases its code to the public under an open source software license, inviting the external community to join the project." West and O'Mahony find both of these models successful ways to initiate an OSS project, each with unique strengths and challenges. In particular, while spinout projects can provide a solid technical foundation for large-scale innovation, the architectural and design goals of the system may frequently remain as undocumented tacit knowledge fully understood only by the original project initiators. As a result, the external community often struggles to develop a sense of ownership and does not benefit from the intrinsic motivation associated with creating a system from the ground up.

Research done by English and Schwiek on OSS divides projects into two phases: initiation and growth [62]. The initiation phase includes tasks that are more commonly associated with closed source or proprietary software development such as requirements gathering, design, initial implementation, and testing. Development is done by a small core group working independently from the community [63]. A first full release of the product represents the transition from the initiation to the growth phase. It is argued that most projects fail to make this transition [62, 64].

The library-specific OSS related literature tends to focus on adoption and contribution to OSS projects. Adopters and contributors are drawn to specific communities associated with OSS projects. In particular, they are distributed in terms of resources (effort, cost) [65, 66] and control that a community offers [67-69]. Libraries benefit from access to open source code in several noteworthy ways including freeing them from vendor lock in, and giving them the ability to customize source code and influence the direction projects take. Adopters wrestle with concerns about how to support installations of OSS, needs for technical expertise, and the hidden costs of having staff spend time supporting, tailoring, and enhancing software [70, 71]. Chudnov states that "the library community is starting to see this pattern play out around library Free/Libre and Open Source Software (FLOSS) applications, with vendors offering support for and integrated services around FLOSS … tools" [49]. Finally, adopters are concerned with the quality of OSS solutions as compared with vended solutions [72-75]. Moore et al. nicely sum up this tension. They say "open source software has often been described as 'free like a puppy,' meaning that even though the applications themselves are free, implementing and maintaining these products requires a long-term investment of time and money, proportional to the complexity of the system [67].”

4.4 Methods

In order to investigate the policies and practices of libraries with regards to software engineering, talent management, and innovation, ARL provided us with access to the raw data from their SPEC Kit 344: Talent Management [76] and SPEC Kit 339: Innovation and R&D [77]. These surveys were distributed to the same libraries as the OSS survey of SPEC Kit 340. The raw data allows us to connect responses from all three surveys to a single respondent. The following Venn diagram shows the number of overlapping respondents.

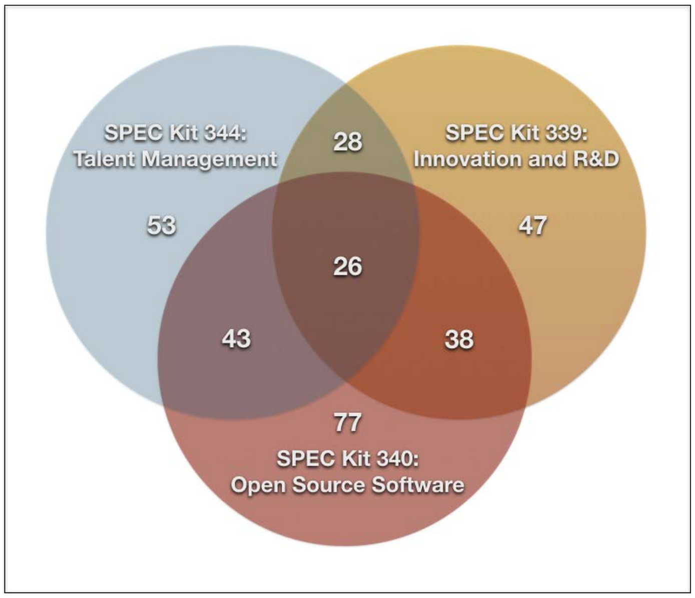

Figure 4.1: Venn diagram showing the overlap of SPEC Kits 339, 340 and 344.

Responses were removed where respondents did not answer all of the questions in one or more of the surveys. Several responses for several of the questions in each of the surveys where discretized before analysis. For example, libraries were asked to indicate the importance of possible criteria they might use when selecting software for purchase or adoption. Responses were presented on a 5 point Likert scale with 1 representing “Not Important” and 5 representing

“Very Important”. Criteria included “staff time to support”, “functionality that best meets our needs”, “control and customizability”, and “staff time to implement”. For our analysis we derived Boolean values for each of these questions with the resulting value being true when the original value on the Likert scale is “Very Important”, and false otherwise.

For each dataset we used logistic regression and step-wise analysis to identify statistically factors correlated with libraries that have code they could release as open source but choose not to.

4.5 Results and Discussion

4.5.1 Software Engineering Policies and Practices

We first looked for correlation in a library’s software engineering policies and practices. Libraries indicating it is very important that deploying purchased or adopted software require minimal staff time were found to be 12.0 times more likely to have software they could release as open source, but chose not to (p-value = 0.0076, R^2 = 0.16).

The OSS SPEC Kit 340 revealed a similar data point. Seventy-seven percent of respondents from this same survey cited concerns around the time commitment to support an OSS community as a factor impacting the decision to open source their software. Libraries working to minimize time spent deploying software would also be concerned about sharing code requiring a time consuming support commitment.

Further, both adoption of and contribution to OSS projects require specialized technical skills [71, 78]. Individuals possessing these skills tend to have a wide variety of responsibilities and be involved with many projects. As such, their time comes at a premium. This may help explain why many institutions who have programmers on staff, have adopted OSS, and even have created their own custom software that could be released as open source choose not to do so in order to avoid the time commitment needed to grow and support an OSS community around the project.

4.5.2 Talent Management Policies and Practices

Analysis of the talent management policies and practices revealed a linear regression model with two significant factors (p-value = 0.0011, $R^2 = 0.28$).

We found that libraries that do not represent employee performance assessment in their strategic plan are 31.8 times more likely to have software they could open source but choose not to (p-value = 0.0030). We speculate that this effect may be explained by looking at the effect these factors have on the culture of an organization. For ease of discussion we state the inverse of our claim: there is a correlation between a library that either cannot share or always shares and a library that represents employee performance assessment in their strategic plan.

As additional background, in the talent management survey of SPEC Kit 344, libraries were asked what talent management activities they currently participate in and which activities are represented in the library's strategic plan. Among the activities listed are professional development opportunities, leadership development opportunities, functional training and employee performance assessment. All activities share a common theme of employee development. In the context of this question employee performance assessment as prescribed in a strategic plan would be used for developmental rather than administrative purposes, meaning that in addition to tracking performance the developmental aspect of employee performance assessments creates a focus and a dialog around the growth and progression of the employee as a professional.

Bettenhausen et al. found evidence supporting their hypothesis that employee performance appraisals used for development were more likely to produce positive outcomes and less likely to produce negative outcomes than appraisals used for administrative purposes [79]. Positive outcomes included “provide quality feedback to recipients”, “give employees a sense of participation in the appraisal system”, “help employees do their jobs better”, “increase productivity of the work unit”, and “increase employees’ feeling of importance to the company”. Negative outcomes included “foster defensive reactions on the part of employees”, “make employees feel vulnerable to retribution”, “create a popularity contest”, and “make employees afraid to tell the truth about coworkers’ performance”. Measuring these outcomes is designed to track overall cultural health of the organization.

From research surrounding general OSS, it is well known that developers who contribute to OSS engender altruistic values and intrinsic motivation [61]. Further, Grant offers evidence that the behavior of those motivated intrinsically and as a result of altruistic values is correlated with greater persistence, performance and productivity [80]. All of these are represented in the outcomes described in the Bettenhausen et al. study mentioned above.

Simply stated, contribution to OSS and strategic, developmental employee performance assessment positively affect the culture of an organization. The presence of both of these factors may represent an organization that is trying to leverage good culture as a strategic advantage.

A second finding related to talent management policies and practices suggests that libraries reporting that cost of living has a neutral or positive impact on recruiting are 10.3 times more likely to have software they could open source, but choose not to (p-value = 0.0062). As before, the inverse statement is easier to understand and will be the basis of our discussion. We found a correlation between libraries that initiate OSS projects when they are able and libraries that report that cost of living has a negative effect on recruiting.

Occupational Employment Statistics from the Bureau of Labor Statistics of the U.S Department of Labor reports that in May of 2015 the average annual salary for Software Developers, Applications (SOC code 151132) is $102,160. The average for the same occupation, but limited to those working in colleges, universities and professional schools, is $77,810 [81]. In other words, Software Developers, Applications in a university setting make 24% less than the industry average.

One explanation for this difference in pay may be found in donative-labor. Becchetti et al. theorized that: “The influential theory of the donative-labour predicts a negative relationship between intrinsic motivations and workers’ pay. The common rationale, consistent with the principle of compensating wage differentials, is that wage-earners will accept lower pay if they find intrinsic (non-monetary) value in their jobs. This implies that intrinsically motivated workers who find that their motivations are satisfied in their occupations and in the missions of their productive organisations, are willing to donate labor to them” [82]. Donative-labor theory is supported by several studies. Preston suggests that workers who are intrinsically motivated view their acceptance of less pay a monetary donation to an organization which produces social benefits [83]. Frank suggests that intrinsic motivations are a form of compensation unto themselves [84]. Rose-Ackerman argues that it is the alignment between workers’ ideals and corporate goals which leads workers to accept lower pay [85]. Finally, Hansmann suggests that this phenomenon acts as a sorting mechanism, by which workers who attach a relatively lower weight to monetary compensation and a relatively higher weight to contributing to the public good are hired in the non-profit industry [86]. Adding specific support to this claim in the context of LIT, Choi, et al. report that altruism and learning are the top two motivations for Library OSS developers — above fun, personal needs, extrinsic rewards and future returns [61].

We suggest that a cost of living high enough to affect recruiting activities acts as a filter by removing qualified candidates from the applicant pool who are not willing to accept a lower wage. Those that remain are prime candidates to be contributors to OSS projects and to drive libraries to initiate OSS projects.

4.5.3 Innovation and R&D Policies and Practices

Analysis of the Innovation and R&D data set revealed that libraries who recognize innovation through press releases are 19.0 times more likely to have software they could open source but choose not to (p-value = 0.0010, $R^2 = 0.25$).

Of the 54 respondents who have a system they choose not to release as open source, 24 (44%) have initiated at least one open source project. This statistic suggests that many libraries who in one case choose not to share their code have in other case(s) released their code as open source.

As previously reported, 74% of those who choose not to open source their code cited “concerns that the code quality is not ready for public adoption.” The perception that the code quality is not acceptable, and therefore cannot be shared, may be related to an organization that is protecting its reputation.

Initiation of or contribution to OSS projects is frequently the result of altruistic values, but may also be related to the reputation or honor of an organization. As Zeitlyn explains: “Software engineers in the open source movement may have sub-groupings which parallel kinship groups such as lineages. Within such groups gift giving is not necessarily or directly reciprocated, instead members work according to the ‘axiom of kinship amity’—direct economic calculation is not appropriate within the group. What Bourdieu calls ‘symbolic capital’ can be used to understand how people work in order to enhance the reputation of themselves and their group” [56].

Recognizing innovation through press releases could be seen as a reputation building activity. Initiating an open source project may be seen similarly, however sharing a system with poor code quality is a possible exception.

4.6 Limitations

Causation is not inferred in any of the reported results. This is purely an observational study. Care must be taken when generalizing survey results to larger populations. The OSS survey SPEC Kit 340 was distributed to 127 ARL member libraries. ARL libraries are often considered a model for best practices, but are not a representative set of research libraries or libraries in general. Further, the 77 respondents of the survey self-selected, introducing bias toward libraries that are interested or invested in OSS. Also, survey fatigue is a significant concern. The OSS survey was relatively long (32 questions), with some questions involving multiple parts and some requiring respondents to look up specific information in order to answer. Several instances were found where respondents did not answer questions completely or did not answer all questions in the survey.

The data used in this analysis comes from surveys performed on human subjects. The human factor introduces variance into the data. This is commonly found in empirical software engineering, just as it is found in other social sciences. The primary implication of this additional variance is that reportable R^2 values tend to be lower than they are in the hard sciences.

4.7 Conclusion

While adoption and contribution are common activities, there are many ARL libraries that have code they could use to initiate an open source software project but have chosen not to. Utilizing data mining techniques to analyze data aggregated from three ARL SPEC Kits we found evidence supporting four findings related to this central idea. The fact that a library could but has chosen not to initiate an OSS project is correlated with the following factors:

1. The library indicates that it is very important that initially customizing and deploying purchased or adopted software require minimal staff time.
2. The library does not represent employee performance assessment in its strategic plan.
3. The library reports that cost of living has a neutral or positive impact on recruiting.
4. The library recognizes innovation through press releases.

For each, we presented research that helps to explain why these correlations make sense within the context of open source software and ARL libraries.

The aggregation of surveys utilized in this paper represent an effort to understand how the culture within a library impacts its propensity to open source their code. In the future, a more granular survey could be designed for this purpose. Gathering data from many more libraries would greatly strengthen these findings.

4.8 Acknowledgements

Thanks to Scott Bertagnole for his insights and for helping to edit this paper. Thanks to Dr. William Lund and the Harold B. Lee Library for supporting this research. We also wish to thank ARL for giving access to the raw SPEC Kit data, making this analysis possible.

Chapter 5

Conclusion

5.1 Summary of Findings

Seventy-seven libraries (61%) responded, and the results were subsequently published as SPEC Kit 340 [8]. Thacker et al. were able to empirically test Askey's assertions, and offered support for Askey's primary claim, as well as many of the contributing factors he identified [45, 46]

The SPEC survey found that 74 respondents (97%) had deployed open source software in their libraries, suggesting that, at least for ARL libraries, adoption of OSS is essentially ubiquitous. The SPEC survey found that 56 respondents (78%) had contributed to one or more open source projects. In contrast to Askey's assertion, we find that initiation of and contribution to OSS projects are, in fact, common practices in libraries. However, we also find that these practices are far from ubiquitous; as Askey suggests, many libraries do have opportunities to initiate OSS projects, but choose not to do so.

Thirty-two (42%) respondents identified themselves as the original developer of an open source project. Respondents initiated an average of 1.4 OSS projects and a median of zero OSS projects. Respondents were asked if any of their in-house software could have been, but had not yet been, released under an open source license. Fifty-two respondents (68%) answered in the affirmative. Further, we find support for only three of Askey's six OSS barriers: time commitment to support the community; code quality is not ready to share; dependence on other systems. Thus, our results confirm many, but not all, of Askey's assertions.

While adoption and contribution are common activities, there are many ARL libraries that have code they could use to initiate an open source software project but have chosen not to. Utilizing data mining techniques to analyze data aggregated from three ARL SPEC Kits we found evidence supporting four findings related to this central idea. The fact that a library could but has chosen not to initiate an OSS project is correlated with the following factors:

- The library indicates that it is very important that initially customizing and deploying purchased or adopted software require minimal staff time.
- The library does not represent employee performance assessment in its strategic plan.
- The library reports that cost of living has a neutral or positive impact on recruiting.
- The library recognizes innovation through press releases.

5.2 Future Work

The sections below describe several areas of research I am interested in that build on the research presented in this thesis.

5.2.1 More Data

The aggregation of surveys utilized in chapter 4 represent an effort to understand how the culture within a library impacts its propensity to open source their code. A more granular survey could be designed for this purpose. Gathering data from many more libraries would greatly strengthen these findings. All findings reported in this paper will require further study before inferences can be made to a broader population.

5.2.2 Reflexivity

The SEQuOIA lab at BYU has published on the topic of reflexivity in OSS. Reflexivity is "the intent of developing software for the benefit of oneself or others like oneself (i.e., for other developers)" [30]. Foushee, et al. found evidence that "the prevalence of reflexivity is positively correlated with success." Future work could include an investigation of reflexivity to discover whether OSS projects created by developers in libraries for developers in libraries are more successful than irreflexive library-related OSS projects.

5.2.3 Project Initiation

West and O'Mahony [87] describe two ways that OSS projects are initiated. Community driven projects are founded and managed within the context of a community. Spinout projects are where "a sponsor of an internally developed software project releases its code to the public under an open source software license, inviting the external community to join the project." West and O'Mahony find both of these models successful ways to initiate an OSS project each with unique challenges. Future work could include an investigation of the success of library-related OSS projects with relation to community driven and spinout initiation models and the challenges related to each.

5.2.4 Commodity Software

van der Linden et al. [88] suggests that "for most products, only a small part (5 to 10 percent) of the software is differentiating (that is, it helps distinguish that product from a competitors' products). This small part provides the added value over the competitors. The remainder is more or less common to the domain, or even across different domains; that is, it's more or less a commodity." They argue that there is a strong case for commodity software to be open source. They further argue that differentiating software should not be open source as this is essentially giving away intellectual property. Future work could look more closely at common challenges and the success of commodity based software vs. differentiating software in the research library context.

References

1. Basili, V.R., R.W. Selby, and D.H. Hutchens, *Experimentation in Software Engineering.* Software Engineering, IEEE Transactions on, 1986(7): p. 733-743.

2. Basili, V.R. *The Role of Experimentation in Software Engineering: Past, Current, and Future*. in *Proceedings of the 18th international conference on Software engineering*. 1996. IEEE Computer Society.

3. Taylor, Q., C. Giraud-Carrier, and C.D. Knutson, *Applications of Data Mining in Software Engineering.* International Journal of Data Analysis Techniques and Strategies, 2010. **2**(3): p. 243-257.

4. Sjoberg, D.I., T. Dyba, and M. Jorgensen. *The Future of Empirical Methods in Software Engineering Research*. in *2007 Future of Software Engineering*. 2007. IEEE Computer Society.

5. Easterbrook, S., et al., *Selecting Empirical Methods for Software Engineering Research*, in *Guide to Advanced Empirical Software Engineering*. 2008, Springer. p. 285-311.

6. Feller, J. and B. Fitzgerald, *Understanding Open Source Software Development*. 2002: Addison-Wesley London.

7. Mockus, A., R.T. Fielding, and J.D. Herbsleb, *Two Case Studies of Open Source Software Development: Apache and Mozilla.* ACM Transactions on Software Engineering and Methodology (TOSEM), 2002. **11**(3): p. 309-346.

8. Bird, C., et al., *An Analysis of the Effect of Code Ownership on Software Quality across Windows, Eclipse, and Firefox.*

9. Krein, J.L., et al. *Language Entropy: A Metric for Characterization of Author Programming Language Distribution*. in *4th Workshop on Public Data about Software Development*. 2009.

10. Bitzer, J., W. Schrettl, and P.J.H. Schröder, *Intrinsic Motivation in Open Source Software Development.* Journal of Comparative Economics, 2007. **35**(1): p. 160-169.

11. Wu, C.-G., J.H. Gerlach, and C.E. Young, *An Empirical Analysis of Open Source Software Developers' Motivations and Continuance Intentions.* Information & Management, 2007. **44**(3): p. 253-262.

12. Lakhani, K. and R. Wolf, *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects.* 2003.

13. Shah, S.K., *Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development.* Management Science, 2006. **52**(7): p. 1000-1014.

14. Ye, Y. and K. Kishida. *Toward an Understanding of the Motivation of Open Source Software Developers*. in *25th International Conference on Software Engineering*. 2003. IEEE.

15. Hars, A. and S. Ou. *Working for Free? Motivations of Participating in Open Source Projects*. in *34th Annual Hawaii International Conference on System Sciences*. 2001. IEEE.

16. Von Krogh, G., S. Spaeth, and K.R. Lakhani, *Community, Joining, and Specialization in Open Source Software Innovation: A Case Study.* Research Policy, 2003. **32**(7): p. 1217-1241.

17. Nagy, D., A.M. Yassin, and A. Bhattacherjee, *Organizational Adoption of Open Source Software: Barriers and Remedies.* Communications of the ACM, 2010. **53**(3): p. 148-151.

18. Wolff-Marting, V., C. Hannebauer, and V. Gruhn. *Patterns for Tearing Down Contribution Barriers to Floss Projects*. in *Intelligent Software Methodologies, Tools and Techniques (SoMeT), 2013 IEEE 12th International Conference on*. 2013. IEEE.

19. Stol, K.-J., et al., *Key Factors for Adopting Inner Source.* ACM Transactions on Software Engineering and Methodology (TOSEM), 2014. **23**(2): p. 18.

20. Bird, C., et al. *Latent Social Structure in Open Source Projects*. in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. 2008. ACM.

21. Crowston, K. and J. Howison, *The Social Structure of Free and Open Source Software Development.* First Monday, 2005. **10**(2).

22. MacLean, A.C., et al., *Knowledge Homogeneity and Specialization in the Apache Http Server Project*, in *Open Source Systems: Grounding Research*. 2011, Springer. p. 106-122.

23. MacLean, A.C. and C.D. Knutson, *Open Source: From Mythos to Meaning.* Salvador, Brazil, 2011: p. 28-41.

24. Gacek, C. and B. Arief, *The Many Meanings of Open Source.* Software, IEEE, 2004. **21**(1): p. 34-40.

25. Chun, S.B., *A Reusable Persistence Framework for Replicating Empirical Studies on Data from Open Source Repositories*. 2011, Brigham Young University.

26. Taylor, Q.C., et al., *An Analysis of Author Contribution Patterns in Eclipse Foundation Project Source Code*, in *Open Source Systems: Grounding Research*. 2011, Springer. p. 269-281.

27. Taylor, Q.C., *Analysis and Characterization of Author Contribution Patterns in Open Source Software Development*. 2012, Brigham Young University.

28. Pratt, L.J., *Cliff Walls: Threats to Validity in Empirical Studies of Open Source Forges.* 2013.

29. Foushee, B.D., *Prevalence of Reflexivity and Its Impact on Success in Open Source Software Development: An Empirical Study*. 2013, Brigham Young University.

30. Foushee, B., et al. *Reflexivity, Raymond, and the Success of Open Source Software Development: A Sourceforge Empirical Study*. in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*. 2013. ACM.

31. MacLean, A.C. and C.D. Knutson. *Apache Commits: Social Network Dataset*. in *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*. 2013. IEEE.

32. Centioli, C., et al., *Open Source Real-Time Operating Systems for Plasma Control at Ftu.* Nuclear Science, IEEE Transactions on, 2004. **51**(3): p. 476-481.

33. Ferrández, J.M., et al. *An Open-Source Real-Time System for Remote Robotic Control Using Neuroblastoma Cultures*. in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. 2010. IEEE.

34. Rahman, M.S., et al. *Implementation of Sctp in an Open Source Real-Time Operating System*. in *Military Communications Conference, 2008. MILCOM 2008. IEEE*. 2008. IEEE.

35. McDonald, C.J., et al., *Open Source Software in Medical Informatics—Why, How and What.* International journal of medical informatics, 2003. **69**(2): p. 175-184.

36. Holland, R.C., et al., *Biojava: An Open-Source Framework for Bioinformatics.* Bioinformatics, 2008. **24**(18): p. 2096-2097.

37. Ratib, O. and A. Rosset, *Open-Source Software in Medical Imaging: Development of Osirix.* International Journal of Computer Assisted Radiology and Surgery, 2006. **1**(4): p. 187-196.

38. van Rooij, S.W., *Adopting Open-Source Software Applications in Us Higher Education: A Cross-Disciplinary Review of the Literature.* Review of Educational Research, 2009. **79**(2): p. 682-701.

39. Dougiamas, M. and P. Taylor. *Moodle: Using Learning Communities to Create an Open Source Course Management System*. in *World conference on educational multimedia, hypermedia and telecommunications*. 2003.

40. O'Hara, K.J. and J.S. Kay, *Open Source Software and Computer Science Education.* Journal of Computing Sciences in Colleges, 2003. **18**(3): p. 1-7.

41. Eyler, P., *Koha: A Gift to Libraries from New Zealand.* Linux Journal, 2003. **2003**(106): p. 1.

42. Anderson, P. *Richard Stallman on the Road Less Travelled*. OSS Watch 2008 11 June 2012 [cited 2015 14 May 2015]; Richard Stallman's views on free and open source software are controversial and quite well known, but what are his views on its use in education? Paul Anderson, from Intelligent Content, catches up with him at the University of Manchester on a rare visit to the UK during Summer 2008.]. Available from: http://oss-watch.ac.uk/resources/stallman.

43. Engard, N.C., *Practical Open Source Software for Libraries*. 2010: Elsevier. 268.

44. Askey, D., *We Love Open Source Software. No, You Can't Have Our Code.* Code4Lib Journal, 2008(5).

45. Thacker, J.C., C.D. Knutson, and M. Dehmlow, *Spec Kit 340*, in *Open Source Software*, L. George, Editor. 2014, Association of Research Libraries: Washington, DC. p. 184.

46. Thacker, J.C. and C.D. Knutson, *Barriers to Initiation of Open Source Software Projects in Libraries.* Code4Lib, 2015(29).

47. Laurent, A.M.S., *Understanding Open Source and Free Software Licensing*. 2004: " O'Reilly Media, Inc.".

48. Kitchenham, B.A. and S.L. Pfleeger, *Personal Opinion Surveys*, in *Guide to Advanced Empirical Software Engineering*. 2008, Springer. p. 63-92.

49. Chudnov, D., *The Future of Floss in Libraries*, in *Information Tomorrow: Reflections on Technology and the Future of Public and Academic Libraries*. 2007, Information Today, Inc: Medford, NJ. p. 19-30.

50. Raymond, E., *The Cathedral and the Bazaar.* Knowledge, Technology & Policy, 1999. **12**(3): p. 23-49.

51. Altman, M., *Open Source Software for Libraries: From {Greenstone} to the {Virtual Data Center} and Beyond.* iassist Quarterly, 2002. **25**.

52. Delone, W.H. and E.R. McLean, *The Delone and Mclean Model of Information Systems Success: A Ten-Year Update.* Journal of management information systems, 2003. **19**(4): p. 9-30.

53. DeLone, W.H. and E.R. McLean. *Information Systems Success Revisited*. in *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*. 2002. IEEE.

54. DeLone, W.H. and E.R. McLean, *Information Systems Success: The Quest for the Dependent Variable.* Information systems research, 1992. **3**(1): p. 60-95.

55. Crowston, K., H. Annabi, and J. Howison. *Defining Open Source Software Project Success*. in *International Conference on Information Systems*. 2003.

56. Zeitlyn, D., *Gift Economies in the Development of Open Source Software: Anthropological Reflections.* Research policy, 2003. **32**(7): p. 1287-1291.

57. Hertel, G., S. Niedner, and S. Herrmann, *Motivation of Software Developers in Open Source Projects: An Internet-Based Survey of Contributors to the Linux Kernel.* Research policy, 2003. **32**(7): p. 1159-1177.

58. Lerner, J. and J. Tirole, *Some Simple Economics of Open Source.* Journal of Industrial Economics, 2002: p. 197-234.

59. Roberts, J.A., I.-H. Hann, and S.A. Slaughter, *Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects.* Management science, 2006. **52**(7): p. 984-999.

60. Dempsey, B.J., et al., *Who Is an Open Source Software Developer?* Communications of the ACM, 2002. **45**(2): p. 67-72.

61. Choi, N. and J.A. Pruett, *The Characteristics and Motivations of Library Open Source Software Developers: An Empirical Study.* Library & Information Science Research, 2015. **37**(2): p. 109-117.

62. English, R. and C.M. Schweik, *Identifying Success and Tragedy of Floss Commons: A Preliminary Classification of Sourceforge.Net Projects.* Emerging Trends in FLOSS Research and Development, 2007. FLOSS'07. First International Workshop on, 2007: p. 11-11.

63. Bergquist, M. and J. Ljungberg, *The Power of Gifts: Organizing Social Relationships in Open Source Communities.* Information Systems Journal, 2001. **11**(4): p. 305-320.

64. Capiluppi, A., P. Lago, and M. Morisio, *Evidences in the Evolution of Os Projects through Changelog Analyses.* 2003.

65. van Rooij, S.W., *Perceptions of Open Source Versus Commercial Software: Is Higher Education Still on the Fence?* Journal of Research on Technology in Education, 2007. **39**(4): p. 433-453.

66. Stranack, K., *The Researcher Software Suite: A Case Study of Library Collaboration and Open Source Software Development.* The Serials Librarian, 2008. **55**(1-2): p. 117-139.

67. Moore, K.B. and C. Greene, *The Search for a New Opac: Selecting an Open Source Discovery Layer.* Serials Review, 2012. **38**(1): p. 24-30.

68. Rafiq, M., *Lis Community's Perceptions Towards Open Source Software Adoption in Libraries.* The International Information & Library Review, 2009. **41**(3): p. 137-145.

69. Kumar, V.J., S, *Adoption and User Perceptions of Koha Library Management System in India.* Annals of Library and Information Studies (ALIS), 2013. **59**(4): p. 223-230.

70. Rehman, A., K. Mahmood, and R. Bhatti. *Free and Open Source Software Movement in Lis Profession in Pakistan*. in *First Open LIS Professionals Conference*. 2011.

71. Muir, S.P., *An Introduction to the Open Source Software Issue.* Library Hi Tech, 2005. **23**(4): p. 465-468.

72. Frumkin, J., *Guest Editorial: Balancing the Playing Field.* Information Technology and Libraries, 2002. **21**(1): p. 2.

73. Chawner, B. *F/Oss in the Library World: An Exploration*. in *ACM SIGSOFT Software Engineering Notes*. 2005. ACM.

74. Breeding, M., *The Viability of Open Source Ils.* Bulletin of the American Society for Information Science and Technology, 2009. **35**(2): p. 20-25.

75. Jaffe, L.D. and G. Careaga, *Standing up for Open Source.* Library Philosophy and Practice, 2007. **9**(2): p. 21.

76. Taylor, M.A. and E. Lee, *Spec Kit 344*, in *Talent Management*, L. George, Editor. 2014, Association of Research Libraries: Washington, DC. p. 183.

77. German, L. and B.S. Namachchivaya, *Spec Kit 339*, in *Innovation and R&D*, L. George, Editor. 2013, Association of Research Libraries: Washington, DC. p. 192.

78. Tennant, R., *The Role of Open Source Software.* LIBRARY JOURNAL-NEW YORK-, 2000. **125**(1): p. 36-36.

79. Bettenhausen, K.L. and D.B. Fedor, *Peer and Upward Appraisals a Comparison of Their Benefits and Problems.* Group & Organization Management, 1997. **22**(2): p. 236-263.

80. Grant, A.M., *Does Intrinsic Motivation Fuel the Prosocial Fire? Motivational Synergy in Predicting Persistence, Performance, and Productivity.* Journal of applied psychology, 2008. **93**(1): p. 48.

81. Bureau of Labor Statistics, U.S. Department of Labor, Occupational Employment Statistics. May 2015 [accessed April 12, 2016]; [http://www.bls.gov/oes/].

82. Becchetti, L., S. Castriota, and E.C. Tortia, *Productivity, Wages and Intrinsic Motivations.* Small Business Economics, 2013. **41**(2): p. 379-399.

83. Preston, A.E., *The Nonprofit Worker in a for-Profit World.* Journal of labor economics, 1989: p. 438-463.

84. Frank, R.H., *What Price the Moral High Ground?* Southern Economic Journal, 1996: p. 1-17.

85. Rose-Ackerman, S., *Altruism, Nonprofits, and Economic Theory.* Journal of economic literature, 1996. **34**(2): p. 701-728.

86. Hansmann, H.B., *The Role of Nonprofit Enterprise.* The Yale law journal, 1980. **89**(5): p. 835-901.

87. West, J. and S. O'Mahony. *Contrasting Community Building in Sponsored and Community Founded Open Source Projects*. in *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*. 2005. IEEE.

88. Van der Linden, F., B. Lundell, and P. Marttiin, *Commodification of Industrial Software: A Case for Open Source.* Software, IEEE, 2009. **26**(4): p. 77-83.