Chapter 5

Political and Social Dimensions of Library Code

The original concept of this report encompassed only code samples and analyses and learn-to-code resources. However, survey responses discussed the political and social dimensions of library code so often as to make them inseparable from the technical dimensions.

Sometimes, this was positive. Matt Weaver's digital signage code (chapter 4) "rescued a rather expensive, and unpopular project"; along the way, he "learned a lot about the emotional impact a technology project can have across staff." Coral Sheldon-Hess's RSS-cache code (chapter 4), which enabled the library to display its diverse social media presences on its home page, incentivized staff members to write more social media content because they were excited to see their new material on top. Hillel Arnold's Captain's Log was written specifically to solve a communication problem, giving staff from different reading rooms an easy way to leave each other notes.

Hillel Arnold's Captain's Log
https://github.com/RockefellerArchiveCenter/captains-log

Respondents wrote of positive emotional impacts on themselves, too. Evviva Weinraub Lajoie discovered "I was capable of building something that thousands of people across the world use to access electronic resources, which was really quite powerful and empowering for me." Several people wrote of their pleasure when their code or documentation helped coworkers to advance their own skills. Jeremy Darrington (chapter 4) said, "I like that coding makes me feel that I'm not helpless, that I can solve some of the problems I face with tools at my disposal."

On the other hand, not all emotional responses were so positive. Many library coders spend a significant amount of time trying to cultivate buy-in, educate their colleagues about technology, or work against siloed organizational structures as they produce inherently cross-departmental work. Code can challenge hierarchies and change workflows, leading to resistance. And, as one librarian writes, "there are folks out there who will hold on to their assumptions about how patrons use library tools no matter what data you show them. (And a corollary, if your data goes against assumptions that are necessary for the survival of a way of thinking or a business, look out. Folks will get NASTY.)"

Coding in libraries often requires the political skills to generate buy-in, surmount institutional barriers, and navigate relationships with management who don't understand what you do. Managers who do understand, or are sympathetic to, coding may face similar challenges on their supervisees' behalf. This chapter outlines issues respondents faced and techniques they used to support and advocate for their projects.

Library Coders' Job Descriptions and Realities

One complication for many library coders is that their job descriptions don't necessarily involve coding. They may have duties that can be achieved far more quickly and effectively with code than by traditional means, or indeed that require at least occasional code editing to be accomplished, but coding is nowhere in the job description. As Carrie Preston puts it, "Certainly my supervisors in my earliest positions never conceived of

my job as being 'about coding,' and I think my activities remained largely mysterious and unfathomable in their eyes."

In some cases, this can make professional development and managerial support hard to come by, even when management recognizes the quality of employees' output. Other librarians, like Angela Galvan, find that "My job description and what I actually do all day are increasingly disconnected things." This may result in a tacit, laissez-faire kind of support, as long as the required work is getting done somehow. On the other hand, a substantial minority of librarians surveyed found that coding became an official part of their jobs, incorporated into subsequent job descriptions, as management recognized its value. For example, Carrie Preston found that "eventually some other members of the cataloging department began to use some of the scripts I wrote, and batch editing and batch loading of bibliographic data (which often involves some coding) did become a formal job responsibility." Josh Westgard is now in a job that is about half coding because he "advocated for the automation of many previously manual tasks."

Across the board, librarians with tech-savvy managers had an easier time getting support for their coding activities (whether formal, like courses, or informal, like time to code at the office as long as the work got done). While many librarians did not indicate whether their managers also had coding skills, 100 percent of those who said their managers were tech-savvy also said they had received some professional development support. Similarly, 100 percent of the coding librarians who are also managers mentioned offering professional development support for coding skills to their supervisees. Indeed, several respondents who are not managers create and run technology workshops for their coworkers.

Buy-in

One issue that came up frequently was buy-in. Although library coders are often solo, and individuals can do a lot with code, it's hard to turn code into a useful service for the library without cooperation. Access to testing and deployment servers, authority over website content, and time for developing and maintaining projects all need institutional support. Numerous respondents talked about both strategies for gaining that support and limitations when they didn't get it.

Bohyun Kim recommended Tito Sierra's exceptionally useful Project One-Pager. This is a document written collaboratively by stakeholders in order to come to a shared understanding of a project. It specifies key information like project scope (including what's out of scope), deadlines, and participants. Not only does this shared understanding promote buy-in, but it also helps everyone see when a project is finished and get the morale boost that comes along with successful project completion.

Project One-Pager
www.slideshare.net/tsierra/the-projectonepager

Coral Sheldon-Hess has also achieved buy-in through documentation. She worked with the web team to write up guiding principles for web design, content, and process.[1] Through researching this document, her team reached a shared understanding of best practices; by writing them down, they generated a reference point for the library as a whole. Sheldon-Hess shared her thoughts on this process in a 2013 LITA Forum presentation.[2]

Documentation can be useful for buy-in throughout a project life cycle, too. Terry Brady notes that it "can allow users to learn at their own pace and to revisit the documentation as often as needed. This is a great approach to achieving buy-in for a solution."

Other respondents achieved buy-in through directly demonstrating the value of library code. Robin Camille Davis did a live coding demonstration of her EZproxy script (chapter 3), and "the people I was with at that demonstration (the systems librarian and the systems manager) were very impressed and got that 'We can do ANYTHING with Python!' gleam in their eyes." Other respondents recommended pilot projects. Often it's hard to talk about what code can do in the abstract, but people respond strongly to prototypes.

Eric Phetteplace (chapter 4) found that his code let his library do a better job of demonstrating its value on campus. Once his form validation code ensured that they were collecting sound reference statistics, they could see that 60 percent of their questions were about technical help. This helped the library advocate for its role in computer literacy and challenge assumptions that it dealt only with books.

Several respondents, particularly in technical services, were able to make strong arguments about the time-saving value of code. We saw in chapter 2 that Becky Yoose saved one to two weeks of cataloger time every year by scripting a repetitive task. Similarly, Carrie Preston noted that "as my department's then-only regular user of [OCLC Macro Language] scripts, I had several times the cataloging productivity of any other cataloger in that department, even while spending a smaller percentage of my time on cataloging." And Annie Glerum (chapter 2) found "that even with reduced staffing, it is possible to achieve both quality and timeliness."

And, when all else fails, some coders go rogue. One noted, "I have learned intentionally breaking systems

known to be fragile is a good way for me to gain the permissions I need to do the work I'd like." Of course, it's always better if the library administration and IT are on board! But coders are by definition inclined to make (and break) things; they tend to find places to exercise their skills or grow deeply frustrated if they can't. One respondent was irked that his code, which made it easier for website users to access digital content, had limited impact because of inadequate support for digitization. He "learned that the impact of code can be limited by administrative lack of resolve, understanding, and focus." And at least eight of the fifty-three survey respondents have changed jobs between answering the survey in spring 2014 and this writing in November 2014. While their reasons vary, this does point to how hard it is to keep coders satisfied if they don't have scope for building things.

Finally, several respondents raised the issue of mission-criticality, but without agreement. Some said that coding mission-critical projects is a good way to achieve buy-in and sustain motivation; others noted that working on key projects is a good way to justify professional development support. However, as Becky Yoose says, "Do not start coding on a project that's mission-critical because that is a good way to fail." She and others recommended building small pilot projects to demonstrate value and build skills before tackling critical services.

Institutional Barriers

Many librarians were missing some important kind of institutional support for learning and writing code. These missing pieces fell into three broad categories:

- lack of support for learning
- lack of support for doing the work
- lack of collaboration

One librarian who hadn't received support for learning to code said, "Coding is really useful, but you're just supposed to know it." Many respondents reported learning to code on their own time, outside of work. Some librarians had difficulty convincing employers to let them spend professional development funds on code learning; indeed, one manager could not secure support for a supervisee because the higher-ups "didn't want her to learn because that would mean that they would have to bump her up a classification level." Other librarians simply didn't have enough professional development funds to cover high-impact learning opportunities like formal classes or conferences. In many cases, the best form of support described was benign neglect—managers who didn't know what these librarians were doing but wouldn't stop them from coding as long as things got done somehow.

> "The institution, however, only gives me $500 in professional development funds per year so although the resources are here to learn whatever I want, any structured learning I want to do comes out of pocket. As it is, the institution is not paying for me to speak at conferences related to my job directly unless they are planned for 12+ months in advance, and I do not have the time to play institutional Calvin Ball with a budget office that doesn't know how libraries work."

Other librarians who already have the skills to code described environments that were hostile to doing that sort of work. Lack of access to servers or permission to install software is a recurrent problem; one librarian says, "I mean, seriously, there is one section where I parse XML with regular expressions. But at the time I didn't have access to install libxml on the system!" Another librarian, whose resume is code-heavy and who was hired in a systems role, found that his managers expected him to use only proprietary software, even when open source options (which he had the expertise to implement) would have been better or cheaper. They also expected him to call vendor support rather than figuring out problems on his own. In one extreme case, a librarian who spends well over half his time on coding and related tasks is at an institution where most units (including his) are explicitly banned from touching code. His middle management recognizes how valuable his work is and finds ways to protect the time while keeping upper management in the dark.

Unsurprisingly, isolation is a major issue for many coding librarians. They may be the only ones in their department, or even their library, who know how to code. Organizational and cultural barriers may prevent them from collaborating with IT or with librarians in other institutions. This is particularly unfortunate because, contrary to popular stereotypes, coding is a profoundly social occupation. Most programs of any size are written by teams; most learning takes place through shoulder-surfing, code review, and other forms of pair programming or mentorship. This is especially true for advanced programming skills, like making good decisions about the overall organization of programs, and for everyday craft knowledge, like discovering good editing and debugging tools.

One librarian wrote, "We had a systems librarian who was very much the fabled hardcore geek of yore, who had basically single-handedly programmed much of the infrastructure we depended on (e.g., ERMS, website CMS, etc.) but was known to only work on a problem if he believed it to be important (not many external suggestions—even from the [University Librarian]—passed this test)." There are good reasons for people to be territorial about code—it's important to have high standards of quality and maintainability for

mission-critical applications—but at this extreme, the whole institution is held hostage because only one person understands the code. The respondent taught himself enough code to solve some problems that the systems librarian wasn't interested in fixing, but this was an enormous lost opportunity for knowledge transfer. Furthermore, since he is self-taught, he recognizes that he doesn't "have any of the best practices that make code sharing easier." This, in turn, will make it harder to collaborate with any future coding coworkers.

Of course, many librarians who code do not have even one coworker they can talk to about code. In their case, the ability to share code and participate in open-source projects is critical for skills development. Many libraries, however, do not have formal policies on whether code can be shared and may not have an informal consensus; some are actively hostile to open source. Dale Askey outlined diverse reasons for this hostility, including perfectionism, fear of ongoing support responsibilities, and misunderstanding of open source.[3] The upshot, however, is untold wasted hours of duplicated work and limits on librarians' ability to increase their own skills.

Bohyun Kim (who ran into this challenge herself) recommends thinking about open source and intellectual property from the very start. Coders are often in fairly junior roles and may not have the ability to negotiate with their institutions; however, it's good to identify what approvals you would need to release your code and who owns it. If you can identify, or create, a release procedure, your code will be more useful and personally rewarding.

Notes

1. Anna Bjartmarsdottir et al., "Plan for the Web Presence," UAA/APU Consortium Library, November 10, 2013, http://connect.ala.org/node/213992.
2. Coral Sheldon-Hess, "Getting Buy-in on User Centricity," presentation. LITA Forum, Louisville, KY, November 7–10, 2013, www.slideshare.net/csheldonhess/lita-forum.
3. Dale Askey, "Column: We Love Open Source Software. No, You Can't Have Our Code," *Code4Lib Journal*, no. 5 (December 15, 2008), http://journal.code4lib.org/articles/527.