DJIM Dalhousie Journal of Interdisciplinary Management

Volume 12 – Spring 2016
djim.management.dal.ca
doi: 10.5931/djim.v12.i1.6450

Librarians and Computer Programming: understanding the role of programming within the profession of librarianship

Abstract: Computer programming is increasingly being discussed as a practice within librarianship. However, contemporary discussions about the role of coding within librarianship often suggest that librarians should or should not learn to code, while failing to qualify how and why librarians are employing code in a professional capacity. By investigating case studies that describe librarians writing code, and literature that describes its historical emergence, this paper qualifies popular discussions of code and librarianship with how and why programming is being used in practice by librarians. These case studies reveal that librarians are writing code for data processing and web services as an extension of their normal responsibilities. This article discusses concerns surrounding the scalability and security of librarians who code professionally, as well as concerns about librarians developing inefficient and insecure software. This article concludes that discussion of the practice of software engineering within librarianship is more useful than arguments over whether librarians should

About the Author: Domenic Rosati is a Master of Library and Information Studies candidate (2017). He is currently involved with several digital projects and initiatives, and works as an Archives Intern at the Dalhousie University Archives and Special Collections. He comes from a background in history and computer science and has a strong passion for digital collection and metadata design, discovery, and administration. His primary research concerns are semantic discovery of heritage resources and bibliographic information including interoperability and potential applications of semantic web technologies within library and archive settings.

Introduction

Increasing attention is being paid to computer programming as an activity within librarianship. Contemporary job postings for librarians are including programming knowledge as a qualification (McGlone, 2013). Communities, such as Code4Lib and its accompanying conference and journal, have grown to support librarians engaging in this kind of work. Sparked by claims that 'all librarians or library students should learn to code,' or 'coding is a fundamental literacy,' contemporary discussions about the role of code within librarianship are growing. However, these discussions often advocate learning computer programming with very little qualification about how this skill is being applied in libraries by librarians. Librarians, library administrators, and library students looking to understand the reasons for coding in librarianship are often met with answers that do not address the historical and societal reasons for why librarians, rather than professional programmers, are taking on this work. These answers also fail to address the practical realities of coding as it is employed professionally by librarians.

As a response, this paper will present the major topics raised in popular discussions of coding in librarianship, which are often framed around the question 'should librarians learn to code,' and will supplement them first by investigating the historical reasons that librarians began programming, as found in librarianship literature, and second by analyzing the realities of librarians who engage in computer programming by looking at case studies. This investigation found three primary reasons why librarians are responsible for programming as part of their jobs: technological changes, such as the emergence of dynamic web applications; the volume of data that librarians need to process, transform, and migrate; and the lack of IT staff, support, resources, and time. Despite an increased responsibility for librarians to develop programming solutions, a key finding in these case studies is that most programming projects are not instances of librarians taking on traditional IT work. Instead, the following discussion will illustrate how programming solutions should be thought of as extensions of the services and work that already fall within the scope of librarianship. In addition, the case studies reveal that the term "programming" within librarianship is used to signify varying levels of proficiency, from knowing how to find and run a script to writing a comprehensive web service or software solution. Finally, the case studies attribute success in coding projects as the ability to identify when programming is an appropriate library solution, and the ability to plan that solution's implementation in a sustainable way, rather than attributing success to technical proficiency with programming languages.

The issues found in the case studies are often not concerned with programming proficiency or learning code but instead of how to design solutions that are efficient,

secure, maintainable, and that can operate on different scales of use. These concerns are traditionally addressed by software engineering. Yet, very few case studies directly discuss these software engineering practices. In light of these issues, this paper contends that discussions of 'should librarians learn software engineering' are more useful than 'should librarians learn code,' which neglects the more pressing issues and realities that are faced by librarians professionally involved with coding. In this article, software engineering is defined as the competency of ensuring maintainable, secure, and sustainable web and software applications through testing, documentation, code style, design, and other best practices. Coding or programming refers to working with programming languages that can handle variables, conditional statements, and looping, such as Python, PHP, and JavaScript. It does not refer to markup, styling, or query languages.

Why Librarians Should Learn Code

A variety of topics are raised in the popular discussions of coding in librarianship. While these discussions have existed at least since 1977, this paper will focus on conversations from the year 2000 onwards, that are representative of a specific and current set of topics. Authors often claim that coding is a useful skill for librarians or library students, but often provide little or no follow through on how those groups might apply coding in a professional context. Several authors justify their advocacy of librarians learning code with the argument that coding is a basic and fundamental literacy and skill (Wisniewski, 2012). This argument is related to the 'Code Year' phenomena in 2012 that saw people, including prominent figures such as New York's mayor Michael Bloomberg commit to learning code through a weekly JavaScript lesson provided by Code Academy. Librarians began the American Library Association's (ALA) Library Code Year Interest Group as a commitment to this 'Code Year' program. Part of the premise of the 'Code Year' is that learning to code is good because coding is a fundamental literacy alongside reading, writing, and arithmetic, and it teaches fundamental skills not acquired elsewhere (Wisniewski, 2012). This is carried over in the overall aims of the ALA Code Year initiatives that sought to support librarians who wanted to learn the fundamentals of programming using the Code Year framework. While coding as a fundamental literacy might explain why coding is beneficial for the individual, it does not explain how coding would look within librarianship.

The problem with the above example is that it frames learning code as a voluntary skill that adds potential value to a librarian's work, without explaining why coding may be a necessity in librarians' roles. Another example of this is the blog post "Should

librarians learn to code?" by Laurie Putnam (2013). She argues that the question is virtually the same as 'should profession X learn to code", because to her, this is a choice of the individual professional and their interest in coding. She does offer some examples of when a librarian might employ code, such as to build a library app, but she does not offer insight into what is unique to the profession of librarianship that might require coding skills, and why coding might be part of a librarian's job description.

Another topic brought up in these discussions is that by employing code and developing applications, librarians can add value to their library's services. As Kim (2012) mentions, "computer-programming skills can also make it possible to create and provide a completely new type of service that didn't exist before" (para. 1). In a similar vein, Yelton (2012) explains that one of the reasons that librarians should learn code is empowerment and creativity. While creating value-added services is certainly a useful and empowering thing librarians could do as part of their service, this discussion fails to frame programming as emerging from real needs-based innovation and continues to portray programming as the voluntary application of on-the-side skills.

Often, communication is cited as a key reason why librarians should learn to code (see Kim, 2012; and Yelton, 2012). For Yelton (2012), one of the reasons librarians should learn to program is because "[l]ibraries need to work well with IT and get good service from software vendors" (p. 5). She continues by saying that learning to code will help librarians have the insight into what is possible with coding, how long a project might take, and how to ask for what you need from the developer. Here, it is thought that better collaboration with IT, or relationships with vendors, are made through librarians learning IT's language. Additionally, Yelton (2013) mentions that librarians should learn to code in order to communicate with other librarians who are engaging with coding. Yelton (2013) says that coding is a social process, and librarians shouldered with coding responsibilities would benefit from a colleague who understands their work.

The discussions that support librarians' learning of code also include more practical reasons offered for learning to code, and these are tied to the professional work of librarians. One author mentions that data processing and migration can be automated with programming, which improves the workflow of librarians (McElfresh, 2012). Another author remarks that customizing and maintaining websites and library applications can be an important part of a librarian's job (Kim, 2012). With programming, the librarian can improve the user experience of these websites and applications (Yelton, 2012). Central to many of these conversations is that librarians are

not receiving institutional and professional support for the programming work they do or that they wish to do (Yelton, 2015). For the purposes of the discussion below, it is important to note that the writings mentioned in this paragraph fail to mention why librarians might be required to implement custom web services or data processing automation. They also fail to mention why these services would be the responsibility of librarians, rather than IT staff or vendors.

Why Librarians Shouldn't Learn Code

Not all people involved in these discussions agree that programming should play a role within librarianship. Reflecting on some of the topics outlined above, Bivens-Tatum (2013) tells of his early experiences learning HTML in library school, with the consideration that he would be applying his knowledge professionally. He found that he never used his knowledge of HTML beyond little modifications to things that he produced in a content management system (CMS). He argues that this example can be applied to librarians wanting to learn code today. He admits that librarians of a technical nature could apply programming to create innovative solutions and services. However, he argues that programming should be entirely optional and that good librarians without technical backgrounds are able to find solutions, or provide similar services, without knowledge of programming (Bivens-Tatum, 2013). Bivens-Tatum (2013) argues that if coding within librarianship is framed as voluntary or value-added, you cannot assert that coding plays, or would play, an essential role within librarianship.

Other writers argue that programming is a potentially dangerous thing for librarians to be involved in. Murray (2012) brings up the potential security implications when librarians are involved in coding. He tells us that when writing code that deals with web services, there are several potential security vulnerabilities and weaknesses that a programmer could open his or her server and data to. He writes of an example where a WordPress website was compromised through its use of custom plugins, and explains that programmers always have to be weary of "code vulnerabilities and attack points" (Murray, 2012, p. 5). Kelley (2015) argues that with only a little knowledge of SQL or Unix commands, common tools which librarians would need to use to deploy their code, a librarian could easily destroy entire library systems. Kelley's (2015) main argument, though, is that librarians who write code would be bad software engineers because, without the deep knowledge of an IT professional, librarians would write "badly engineered and badly tested code" (p.13), which would require significant resources devoted to software up-keep. Some commenters on Kim's (2011) blog react to Kim's advocacy for librarians learning code with similar sentiments. Commenters

argue that treating programming as something one can just pick up and do devalues the work and education of professional programmers. The commenters want librarians to be aware that formal software and web development would demand a certain amount of maintenance and consideration of long-term responsibility. As one commenter puts it, "who would fix [the software or the website] if it's broken? who would fix [it] if it's broken and the said librarian is away for a conference? who would maintain [it] if the said librarian leave[s]?" (Ranti, 2011)

Those critical of librarians coding argue that it is an unfair and unintelligent staffing practice within libraries. Some of these discussions can be found as comments on Kim's (2011) blog post that poses the question of whether librarians are uniquely situated to understand library systems and user requirements. One commenter explains that it devalues the IT professional to say they would not understand the requirements of libraries and library users, because understanding user requirements is part of all good software engineering education and practice (An IT Person, 2011). Another commenter repeats a common sentiment: professional programmers have gone through extensive training that make them competent in their programming abilities (Computer Engineer & Librarian, 2011). To ask librarians to competently perform on the same level as IT professionals without the same training is not an intelligent staffing choice (Will K, 2011). The above statements are related to another argument that libraries tend to require or favour ALA accredited degrees for their programming hires. Commenters note that this is an unfair practice which favours less competent people because of the notion that they can understand libraries better; one commenter likens the practice to requiring the library's lawyers to all have ALA accredited library degrees (Will K, 2011).

Should Librarians Learn Code?

In response to these conversations, Wilkinson (2013) introduces a conceptual schema with which we can understand the positions taken on whether or not librarians should learn code. He says that within the conversation there is strong essentialism, which states that all librarians should learn to code, and weak essentialism, which states that all libraries need someone to code. The strong essential approach encompasses those discussions which advocate all librarians learning code due to the topics illustrated above. In addition to these, Wilkinson (2013) identifies two arguments within strong essentialism. The maintenance argument states that librarians need to code in order to maintain library systems and applications, and the forward thinking argument states that librarians need to be able to code in order to innovate and provide cutting edge services.

Wilkinson (2013) explains that the weak essential approach, which states all libraries need somebody to code, actually applies to any discussion on whether a particular skill in librarianship is essential (Wilkinson, 2013). Reframing the argument so that programming is an essential skill within the library, but not necessarily for all librarians, is a useful way of approaching programming in librarianship. By reframing the argument in this way, it puts running the library as the starting point to discussing the why and how of programming in libraries. However, since only all libraries and not all librarians need programming, the argument leaves open the possibility that libraries could just as well hire and employ only IT professionals for programming. Both the strong and weak essential approach fail to explain the nature of programming within librarianship because they do not provide an explanation for why librarians would be hired, tasked, and responsible for programming, instead of an IT professional.

In librarianship competency and job positing analysis literature, scholars have produced a large body of literature on competencies within librarianship. However, these scholars have not produced concrete evidence of what the competencies of programming mean or look like (see Raju, 2014 for an analysis of digital competencies literature). Additionally, these scholars have a tendency to celebrate programming languages in librarianship without qualifying how and why those programming languages will be employed (Choi & Rasmussen, 2006). In an analysis of frequencies of coding languages within librarianship, Maceli (2015) found that JavaScript and PHP were the top programming languages included in job posts for developers and librarians on Code4Lib's job posting site. However, this analysis tells us very little about how these languages will be used by librarians, or what would be expected from the successful candidate, other than that the projects will be web based. These languages can be used for simply modifying or integrating services, or for developing whole web applications.

Historical Origins of Librarians in Code

Previous arguments, discussions, and frameworks lack the explanative power to identify why librarians are required to learn code and are fulfilling certain programming functions within libraries. The following historical discussion aims to answer the whys of coding in librarianship. The earliest of these discussions can be found in librarians' hands-on involvement in automation. Before the development of web services in the late 1990s, the majority of programming work in libraries was devoted to library systems such as the electronic catalogue (Rhyno, 2003). Rhyno (2003) notes that instances of librarians' involvement with programming were usually related to managing in-house developments, as opposed to working with vendor

solutions. Vendors advocated that librarians should stay out of programming and that better relationships with the vendor should be developed instead (Rosenberg, 1987). However, since library systems were more difficult to customize, Rhyno (2003) notes that librarians with IT responsibilities were not programmers but translators between the vendor, IT, and library administration. Davis (1977) noted that computer programming in librarianship in the 1970s was about communication with programming and automation professionals who were being employed to provide automation solutions in the library. The motivation for librarians learning to code was to provide some measure of independence from IT and vendors (Davis, 1977). Like the authors of contemporary popular discussions, Davis (1977) also noted the potential for creative programming work or the empowerment of librarians to fix systems themselves as value-added benefits of learning programming.

Library Hi Tech Volume 21, Issue 3, (2003) features several articles about the historical emergence of the systems librarian. In one of these articles, Rhyno (2003) tells us that the genesis of programming within librarianship really lies in the common gateway interface (CGI) protocol that allowed applications and databases to be connected to websites dynamically. Librarians could now use CGI scripting languages such as Perl, known as the 'Swiss army knife' of scripting within librarianship, and later PHP, to integrate various library applications, databases, and content into the web. Integrating, creating, or maintaining applications and databases with scripting was much easier and maintainable than customizing larger self-contained library systems, which required the use of lower level languages and programming concepts such as threading and memory handling. Some of the early coding projects which librarians worked on were the first public web integrations (as online public access catalogues, or OPACs) with the library's integrated library systems (ILS) (Rhyno, 2003). Scripting and web languages allowed librarians to take control and provide web services that they would have had to wait for vendors or in-house staff to implement. While systems librarianship had existed since the genesis of library systems and automation, the responsibilities of systems librarians now began to encompass more direct programming roles. As more and more services and systems utilized or allowed these scripting languages and dynamic web services, systems librarians began to also be more involved in the maintenance and implementation of IT solutions.

Kelley (2015) provides a more critical perspective on the origins of the system librarian, particularly as the role relates to programming. He argues that the origin of systems librarianship is actually due to IT staff shortages, or hiring librarians where programmers should have been employed. As a result, Kelley (2015) says that systems librarians should be thought of as IT imposters lacking the depth and knowledge which would

belong to professional programmers. He suggests that librarians involved with programming should make room for more IT professionals in libraries and relegate their work to communication or project management roles. While others have confirmed that staff shortages do contribute to librarians' involvement in programming (see Watson, Hawthorne & Wishnetsky, 2008; Randtke, 2013), not all authors see the role of the systems librarian as detrimental. In his article from Volume 21, Issue 3, of *Library Hi Tech*, Goddard (2003) argues that while systems librarians may lack programming skills, they add communication, project management, and instruction or training skills to library systems and programming projects.

In addition to the systems librarian, the emergence of the web services librarian also tells of how librarians began to be involved in, and responsible for, programming. Roberts (2005) tells us that instances of librarians involved in programming usually arose from wanting to connect a database or collection to a front end website as an online service. As in the case of systems librarians, server-side scripting was seen as a new window into creating library web services (Nackerud, 1998). As dynamic web applications and services that included scripts and connections to databases became more common, the programming role of librarians expanded. Librarians responsible for previously static web services using markup languages often had to be involved with scripting and modifying existing code in order to make web services more accessible and debug errors in website behaviour (Marchant, 1999). With the increasing popularity of using the web as a platform for delivering services, librarians were tasked with more responsibilities for web services, and needed to use code to fulfill their emerging role (Tidal, 2012). With CMSs and application programming interfaces (APIs), dynamic web services became more powerful and easier to use, with the possibility that web services could be developed with much less programming knowledge. However, integrating, customizing, and implementing new web services, such as implementing services into the OPAC, still requires programming knowledge in most cases (see McMullen & Gray, 2012).

While job postings for the emerging role of the digital librarian, as steward of digital collections and data, often list programming knowledge as a requirement for this position, a unique role as it relates to programming is rarely described. Instead, cases of digital projects librarians programming are cases of web projects involving data processing, migration, and automation. The latter type of programming is even more common in the role of the metadata librarian. One metadata librarian explained that part of the professional transformation from cataloguing librarian to metadata librarian could be characterized by the use programming languages for automation and metadata processing (Schwartz, 2010). With the high volume of metadata produced,

or needing to be transformed and re-fitted, Finch (2013) argues that data transformation and migration is the most important work of metadata librarians, and this work must be automated with programming. Intner and Lazinger (2010) explain that this is the reason why entry-level cataloguers are being asked to have knowledge of programming languages.

Case studies

In order to further supplement discussion of the how and why of librarians programming, the following case studies will present several instances found in librarianship literature where librarians were actively writing their own code and solving problems in the two areas discussed above: data processing and web services. One of the most common instances where librarians developed programming solutions was for data processing. These data processing tasks include the migration, transformation, ingestion, and extraction of data. Each of these cases identifies programming as a solution because of the need for large amount of records to be processed within a manageable workflow. In these cases, authors mention that batch processing tools such as MarcEdit and Open Refine could be employed as lesser alternatives to programming (Frank, 2013). Others mention that programming is an optional but beneficial solution needed for effective workflow (Godfrey & Kenyon, 2015). Still more explain that many data processing solutions have not yet been developed, or certain applications, services, and library systems require programming in order to perform data processing and migration tasks (Rimkus & Hess, 2014). The scope of programming solutions developed includes developing scripts for remediating metadata (Rimkus & Hess, 2014), ingesting metadata and records into software with various constraints (Abraham, Chapman, Flecker, Kreigsman, Marinus, McGath & Wendler, 2005), and automating metadata creation and updating (Randtke, 2013). These initiatives confirm the conditions described above about the historical emergence of programming within metadata librarianship, which shows that working with metadata has required librarians to write code.

Several of the above case studies deal with migration and transformation of MARC records. They illustrate the why and how of cataloguers employing code. In general, there are two reasons brought up in these case studies about cataloguers programming: 1) programmers do not understand MARC records well enough to provide adequate solutions for librarians; and 2) scripting enables staff to have a more manageable MARC record workflow. Thomale (2010) argues that programmers have a hard time understanding the subtleties of MARC and writing programs that can adequately interpret MARC records. Frank (2013) notes that not only should

cataloguers learn code to communicate better with programmers to develop solutions for MARC records, but cataloguers need to learn to code their own solutions for processing MARC records. Frank's article goes on to describe how he employed Python scripting, in conjunction with other MARC processing tools, in order to fix MARC records as they were exported by Archives Toolkit, which would otherwise have had to be fixed by hand.

Another two articles document the creation of Perl scripts for processing and ingesting MARC records (Highsmith, Jordan, Llona, Murray & Summers, 2002; Surratt & Hill, 2004). These cases illustrate the formal collaboration between IT professionals and librarians to develop a maintainable library solution to simplify working with MARC records. For Highsmith et al. (2002), their script library was developed for librarians by librarians in order to batch migrate or modify MARC, as is shown in various case studies found in the article. Surratt and Hill (2004) are a librarian and developer team who worked on a script to solve the cumbersome workflow of creating MARC records from electronic theses & dissertations (ETDs). Surrat and Hill (2004) addressed the issues of programmers not understanding MARC as their article describes a workflow where the librarian was responsible for programming the part of the script related to metadata, and the developer was responsible for the rest.

The other major area in which librarians were found programming in these case studies was in the development of web services. In addition to the power that server-side scripting gave librarians over developing better web services (as discussed above), two other areas can be found in which librarians were programming web services: integrating and using APIs, and developing client-side services using front-end programming languages such as JavaScript. In contrast to the automation scripts employed for data processing explored above, these projects included a wider range of what was being developed, at what level, and why. McMullen and Gray (2012) mention that the most common web services involved in programming projects are the integration of services with the OPAC. They mention that librarians responsible for managing OPACs have now become responsible for integrating services into or from the OPACs through programming. For their project, the librarians McMullen and Gray (2012) considered the cumbersome workflow for keeping faculty up to date on the acquisitions they requested. After identifying the problem, they describe the library widget they developed that displayed departmental book acquisitions in real time. They discuss their use of PHP and JavaScript to retrieve data from the library OPAC and display it on a website. Other projects mention developing Chrome extensions (Schulkins & Schulkins, 2015), implementing APIs that require modifying scripts for local use (Neugebauer, Carson & Krujelskis, 2015), and adding various services to library web

pages using Perl (Bartle, 2000), ASP.Net languages (Greene, 2008), and the JavaScript library JQuery (Miller-Francisco, 2010; Michel & Ladd, 2015).

Beyond the technical reasons for programming explained above, there was significant discussion in these case studies about the professional reasons why librarians were involved in programming. Many librarians mentioned that the necessity for them to learn programming was prompted by shortages of staff, time, or software capabilities. Some created their projects because there was a shortage of money to hire professional programmers for the project (Watson, Hawthorne & Wishnetsky, 2008). For others, librarians took on programming projects because their IT staff would not (Randtke, 2013). Randtke (2013) explains that librarians have to be weary of the time commitments their potential projects might require of IT staff. Randtke (2013) also explains that IT staff may not be willing or able to support programming projects or initiatives, as had happened in her case. McMullen and Gray (2012) mention that, in managing a library's web presence and applications, many web services solutions and responsibilities that involve programming fall on librarians, not on IT staff. Orphanides (2011) speaks to this when he says, of developing a public touchscreen application, "[t]he task of selecting and developing content, identifying and configuring additional hardware, and designing and implementing an interface, fell to me, along with my colleagues Keith Morgan, Principal Librarian for Digital Media, and Jason Walsh, Technology Support Specialist" (p. 4).

Contrary to some authors' reluctance about librarians coding, as discussed above, software engineering is often raised and discussed in these case studies. Belfiore (2012) discusses the potential problems of maintaining code and code-based initiatives when a librarian with code knowledge is replaced by one without. Many of the case studies regarding scripts for data processing mention that while programming is a great solution, lack of programming knowledge can become a barrier when the scripts have to be run or modified by other staff in other situations (Rimkus & Hess, 2014; Frank, 2013). Orphanides (2011) and others reflecting on their projects discuss the need for better software engineering practices and well-written code in order for the project to be sustainable in the long term. In addition, part of Chudnov, Kerchner, Sharma, and Wrubel's (2014) article is aimed at librarians and archivists, to explain how sustainability works in software projects, by explaining versioning, documentation, code repositories, and more. Despite the above concerns and issues, few of these case studies address security, one of the main vulnerabilities that Murray (2012) worries about. Neither do these case studies directly discuss how they dealt with efficiency and scalability in code projects. Importantly, there were no direct discussions about testing web and software projects.

Discussion

How do the practical realities found in the above case studies relate to popular discussions about librarians coding? In some of these discussions, the programming work of a librarian was framed as voluntary. Yet, in these case studies, there were a variety of reasons why programming was considered a necessity by librarians: the project was their responsibility, the librarian was the only one who would be able to do it, there was a shortage of IT professionals, the project needed the librarian's knowledge of metadata standards, and the librarian required programming solutions for transforming and migrating the large amounts of data particular to their situation. Contrary to some of the popular discussions that advocated for the hiring of professional programmers to undertake librarians' programming duties, these projects made sense within the scope of the librarian's responsibilities and, due to the relatively low time commitment of the projects, would have been too expensive or impractical to find IT professionals for. For web services librarians, the projects were usually customizations or new widgets, which were extensions of what librarians might have already been expected to be doing within a CMS or a static web page. Some of the popular discussions tended to frame librarian programming projects as unessential to library functions, yet all of these case studies, even the value-added ones, were grounded in a real need for functionality, workflow, or service.

Perhaps to respond to some misgivings about librarians involved in code, many of the case studies were actually collaborations between developers and librarians, wherein both developers and librarians wrote code and worked on the same project. In popular discussions about librarians and code, communication between IT and librarians was cited as an important reason for librarians engaging in programming. Some of the above case studies illustrated instances of communication that became formal collaboration. Other case studies noted that their work was intended to get librarians communicating and collaborating with programmers regarding metadata (Frank, 2013). For McMullen and Gray (2012), the most important aspect of communicating with IT was remaining on good terms with them while working relatively independently from them. For Rantke (2013), the most important aspect of communicating with IT was learning what IT did not do and did not know.

One of the significant elements noticed in these case studies is that authors mean different things when they talk about programming. While using programming languages and computational thinking is at the core of what they are talking about, there is a very large range of difficulty and skill level demonstrated within these projects. First, when authors talk about programming projects, sometimes they just

mean connecting applications together using web services, content management systems, and APIs with almost no programming involved (Walker, 2007; Banerjee & Johnson, 2015). In several other case studies, authors are just identifying scripts written by others that they employed for their workflow (Zou, 2015; Neugebauer & Han, 2012). These authors refer to scripting and scripts for data processing, but the primary focus is placed on being able to run these scripts (Donnelly, 2014; Neugebauer & Han, 2012). In the above two examples, librarians still need technical literacy to be able to find and understand scripts or APIs. Their technical literacy enables them to provide a solution, and to modify, compile, and run the script for local implementation. In other case studies, librarians were developing their own scripts (Ogier & Aschmann, 2013) or modifying scripts for re-use (Frank, 2013). Even beyond this, librarians are writing their own complete web services (McMullen & Gray, 2012; Greene, 2008), or complete tools for library solutions (Jenkins, 2009). While there were some larger projects described (Chudnov et al, 2015), most of the programming done by librarians was below the levels of managing large-scale library software projects or vital services, that would traditionally be managed by library IT staff. The consideration of different levels of programming done by librarians is important because some of the negative reactions to librarians programming are predicated on thoughts that librarians are involved in larger projects or more vital systems than they actually are. Here, the programming knowledge appropriate to librarians' projects would not require the formal education and skills of IT professionals. These considerations have implications for library administrators, so they can understand what they should and should not expect from librarians, and for library students and librarians in continuing education, to understand what levels of programming they might be asked to engage in. Finally, a grounded understanding of the levels in which programming is employed is much more useful than the competency and skillset literature outlined above, which struggles to explain why and how programming might be employed professionally.

An important factor that exists implicitly in these case studies is the idea of communicating code within the library profession. While popular discussions have considered administrative and professional support for librarians as an important factor in their participation in coding, those discussions have not considered how that support is built up through networks of communicating code among librarians. What is important about the case studies examined above is that they form a body of literature in which programming solutions to a library problem is communicated, and these solutions are communicated through the reading, writing, and sharing of code for re-use. This is unique, because it requires confidence in code literacy on the part of the reader. If the reader is able to make sense of the code, this body of literature plays

a valuable role within librarianship, because it enables librarians to potentially employ shared programming solutions to their library problems, even if that just means the librarian can share the code with IT staff.

Conclusion

This discussion has qualified the debate surrounding whether librarians should learn code by explaining how librarians are coding, as found in case studies and an investigation into why systems, web services, digital projects, and metadata librarians began to code. The main problem with popular discussions on coding in librarianship is that authors do not define what coding within librarianship actually looks like or the reasons why it would be a professional responsibility. Those discussions make programming for librarians seem entirely optional, or suggest that programming work can be taken over by IT staff without complications. An investigation into when librarians first began programming has shown that changes in technology, such as the emergence of the dynamic website and the nature of metadata, have caused new roles for programming within librarianship. In order to have a richer understanding of how and why librarians are coding, case studies of data processing and web services projects where librarians employed code were presented. In the case studies, librarians engaged in programming as extensions of their responsibilities within web and metadata librarianship, as well as due to shortages of IT staff. These case studies demonstrated that framing coding within librarianship as completely voluntary is inadequate: often librarians were engaging in programming because it was their responsibility and, even when the project is value-added, it is still rooted in a real need to solve a library problem with a programming solution. In contrast to the concerns over librarians engaging in IT work, librarians were not engaging in boundary crossing. Their programming work was largely an extension of responsibilities they already had, and the programming work they were doing was at a much simpler level and smaller scale than what professional programmers might be involved in. Contrary to what some commenters might have expected, the concern for the software quality of programming projects was often addressed in these case studies. In addition, these case studies recognized a need for better software engineering practices among librarians. Also, Murray's (2012) concerns over security still remained unaddressed in the case study literature and these case studies did not discuss some of the more technical aspects of application design such as efficiency, speed, and handling different scales of data or users. These considerations suggest that learning code is not the primary obstacle for librarians. Instead, the obstacle is learning and practicing software engineering so that their coding projects are scalable, secure, and well-maintained.

Authors meant quite a range of things when they spoke of programming, from being able to run and modify scripts, to being able to write complex web services, which is perhaps feeding into why some suggest coding as an easy skill to pick up, while others suggest that programming is completely beyond the capabilities of librarians. This has important implications for the popular discussion at large: for competency literature to understand programming, for library administrators to understand what should and should not be expected of librarians, and for library students and librarians to understand the different levels of programming that might be required of them. What is common to this body of programming literature is a unique set of writings that communicate code solutions between librarians. The above case studies indicate that the current challenges involved in librarians doing programming focus more on issues related to software engineering, and less on issues surrounding code and programming. Therefore, discussions of software engineering within librarianship rather than arguments over whether librarians should learn to code would be more beneficial for programming librarians. Today, librarians are involved in a great many open source projects that are widely used as standard library applications, such as the Islandora project, a digital repository framework initiated by librarians. More research is needed to understand how librarians are using and employing code so that a fuller scope of examples can inform the how and why of coding within the profession of librarianship, and so that evidence-based suggestions about programming education and practice can be made.

References

Abraham, S., Chapman, S., Flecker, D., Kreigsman, S., Marinus, J., McGath, G. & Wendler, R. (2005). Harvard's perspective on the archive ingest and handling test. *D-Lib Magazine*, *11*(12), 1. http://doi.org/10.1045/december2005-abrams

An IT Person (2011). On Kim, B. *why not grow coders from the inside of libraries?* [Web log comment]. Retrieved December 14, 2015 from http://www.bohyunkim.net/blog/archives/1099

Banerjee, K. & Johnson, M. (2015). Improving access to archival collections with automated entity extraction. *The Code4Lib Journal*, (29). Retrieved from http://journal.code4lib.org/articles/10726

Bartle, L. (2000). Mounting a web-accessible database: a model for beginners. *College & Undergraduate Libraries*, *7*(2), 111.

Belfiore, D. (2012). Case study: using perl and cgi scripts to automate a quality control workflow for scanned congressional documents. *The Code4Lib Journal*, (17). Retrieved from http://journal.code4lib.org/articles/6731

Bivens-Tatum, W. (2013). *Why I ignore gurus, sherpas, ninjas, mavens, and other sages*. Retrieved from https://blogs.princeton.edu/librarian/2013/03/why-i-ignore-gurus-sherpas-ninjas-mavens-and-other-sages/

Breeding, M. (2002). Expanding the systems librarian's toolkit. *Information Today*, *19*(1), 36.

Choi, Y. & Rasmussen, E. (2006). What is needed to educate future digital librarians: A study of current practice and staffing patterns in academic and research libraries. *D-Lib Magazine*, *12*(9). http://doi.org/10.1045/september2006-choi

Chudnov, D., Kerchner, D., Sharma, A. & Wrubel, L. (2014). Technical challenges in developing software to collect twitter data. *Code4Lib Journal*, (26), 1.

Computer Engineer & Librarian (2011). On Kim, B. *why not grow coders from the inside of libraries?* [Web log comment]. Retrieved December 14, 2015 from http://www.bohyunkim.net/blog/archives/1099

Finch, M. (2013). The evolving metadata librarian: creating and managing data about data. In Peacock, R. & Wurm, J. (Ed). *The New academic librarian: Essays on changing roles and responsibilities*. Jefferson: McFarland.

Frank, H. (2013). Augmenting the cataloger's bag of tricks : using marcedit, python, and pymarc for batch-processing marc records generated from the archivists' toolkit. *The Code4Lib Journal*, (20). Retrieved from http://journal.code4lib.org/articles/8336#note1

Goddard, L. (2003). The integrated librarian: IT in the systems office. *Library Hi Tech*, *21*(3), 280–288.

Godfrey, B. & Kenyon, J. (2015). The geospatial metadata manager's toolbox: three techniques for maintaining records. *The Code4Lib Journal*, (29). Retrieved from http://journal.code4lib.org/articles/10601

Gordon, R.S. (2003). Overcoming the systems librarian imposter syndrome. *Libres*, *13*(2). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-3042638951&partnerID=40&md5=00c4b33783b9e7117db687e3657db647

Greene, A. (2008). Managing subject guides with SQL Server and ASP.Net. *Library Hi Tech*, *26*(2), 213–231.

Highsmith, A., Jordan, M., Llona, E., Murray, P. E. & Summers, E. (2002). MARC it your way: MARC.pm. *Information Technology and Libraries*, *21*(1), 19–25.

Intner, S.S. & Lazinger, S.S. (2011). Afterword. In *Conversations with catalogers in the 21st century*. Santa Barbara, California: Libraries Unlimited.

Jenkins, K. (2009). Deciphering journal abbreviations with jabbr. *The Code4Lib Journal*, (7). Retrieved from http://journal.code4lib.org/articles/1758#fn7

Kane McElfresh, L. (2012). The year of coding, the future of catalogers. *Technicalities*, *32*(4), 5–7.

Kelley, K.J. (2015). *The Myth and Magic of Library Systems*. Chandos Publishing.

Kim, B. (2011). *Why not grow coders from the inside of libraries?*. Retrieved from http://www.bohyunkim.net/blog/archives/1099

Kim, B. (2012). *effectively learning how to code: tips and resources*. Retrieved from http://acrl.ala.org/techconnect/post/effectively-learning-how-to-code-tips-and-resources

Maceli, M. (2015). What technology skills do developers need? A text analysis of job listings in library and information science (LIS) from Jobs.code4lib.org. *Information Technology & Libraries*, *34*(3), 8–21.

McGlone, J. (2013). Looking under the hood: a view of the digital projects librarian in the academic library. In Peacock, R. & Wurm, J. (Ed). *The New academic librarian: Essays on changing roles and responsibilities*. Jefferson: McFarland.

McMullen, A. & Gray, B. (2012). From static to dynamic using the OPAC to generate real-time lists of departmental acquisitions for library current awareness service. *Library Hi Tech*, *30*(4), 673–682. http://doi.org/10.1108/07378831211285121

Michel, J. P. & Ladd, M. (2015). "Snow Fall"-ing special collections and archives. *Journal of Web Librarianship*, *9*(2-3), 121–131. http://doi.org/10.1080/19322909.2015.1044689

Miller-Francisco, E. (2010). Creating Dynamic Websites Using jQuery. *Computers in Libraries*, *30*(6), 26–28.

Murray, P. E. (2012). *The security implications of teaching librarians to program*. Retrieved from http://dltj.org/article/security-implications-of-librarian-developers/

Nackerud, S. A. (1998). The potential of cgi: using pre-built cgi scripts to make interactive web pages. *Information Technology and Libraries*, *17*(4), 222.

Neugebauer, T., Carson, P. & Krujelskis, S. (2015). Using SemanticScuttle for managing lists of recommended resources on a library website. *The Code4Lib Journal*, (27). Retrieved from http://journal.code4lib.org/articles/10269

Neugebauer, T. & Han, B. (2012). Batch ingesting into eprints digital repository software. *Information Technology & Libraries*, *31*(1), 113–125.

Ogier, A. L., Sechler, M. W. & Aschmann, A. (2013). Teaching wild horses to sing: managing the deluge of electronic serials. *Serials Librarian*, *64*(1-4), 99–104. http://doi.org/10.1080/0361526X.2013.760354

Orphanides, A. K. (2011). Lessons in public touchscreen development. *The Code4Lib Journal*, (15). Retrieved from http://journal.code4lib.org/articles/5832

Putnam, L. (2013). *Should librarians learn to code?*. Retrieved from http://www.nextlibraries.org/2013/12/should-librarians-learn-to-code/

Raju, J. (2014). Knowledge and skills for the digital era academic library. *The Journal of academic librarianship*, *40*(2), 163–170. http://doi.org/10.1016/j.acalib.2014.02.007

Randtke, W. (2013). Automated metadata creation: Possibilities and pitfalls. *The Serials librarian*, *64*(1-4), 267–284. http://doi.org/10.1080/0361526X.2013.760286

Ranti. (2011). On Kim, B. *why not grow coders from the inside of libraries?* [Web log comment]. Retrieved December 14, 2015 from http://www.bohyunkim.net/blog/archives/1099

Rhyno, A. (2003). From library systems to mainstream software: How Web technologies are changing the role of the systems librarian. *Library Hi Tech*, *21*(3), 289–296.

Rimkus, K. R. & Hess, K. M. (2014). HathiTrust ingest of locally managed content: a case study from the University of Illinois at Urbana-Champaign. *Code4Lib Journal*, (25), 7.

Roberts, G. (2005). Learning server-side scripting. *Computers in libraries*, *25*(8), 37–39.

Sanchez, E. (2011). *Conversations with catalogers in the 21st century*. Santa Barbara, California: Libraries Unlimited.

Schulkins, D. R. & Schulkins, J. (2015). Streamlining book requests with chrome. *The Code4Lib journal*, (30). Retrieved from http://journal.code4lib.org/articles/10996

Schwartz, C. (2011). Changing mind-set, changing skill set: transition from cataloger to metadata librarian. In *Conversations with catalogers in the 21st century*. Santa Barbara, California: Libraries Unlimited.

Surratt, B. E. & Hill, D. (2004). ETD2MARC: a semiautomated workflow for cataloging electronic theses and dissertations. *Library collections acquisitions & technical services*, *28*(2), 205–223. http://doi.org/10.1016/j.lcats.2004.02.014

Tennant, R. (1999). Skills for the new millennium. *Library journal*, *124*(1), 39.

Thomale, J. (2010). Interpreting MARC: where's the bibliographic data? *The Code4Lib journal*, (11). Retrieved from http://journal.code4lib.org/articles/3832

Tidal, J. (2013). The evolving role of the web librarian. In Peacock, R. & Wurm, J. (Ed). *The New academic librarian: Essays on changing roles and responsibilities*. Jefferson: McFarland.

Walker, D. (2007). Building custom metasearch interfaces and services using the MetaLib X-Server. *Internet Reference Services Quarterly*, *12*(3/4), 325–339.

Watson, J., Hawthorne, D. & Wishnetsky, S. (2008). ERM on a shoestring: betting on an alternative solution. *The Serials Librarian*, *54*(3-4), 245–252. http://doi.org/10.1080/03615260801974206

Wilkinson, L. (2013). Is coding an essential library skill? Retrieved from https://senseandreference.wordpress.com/2013/03/08/is-coding-an-essential-library-skill/

Will K. (2011). On Kim, B. *why not grow coders from the inside of libraries?* [Web log comment]. Retrieved December 14, 2015 from http://www.bohyunkim.net/blog/archives/1099

Wisniewski, J. (2012). Parlez-Vous Code? *Online*, *36*(6), 57–60.

Yelton, A., & ALA TechSource. (2015). *Coding for librarians: learning by example* (Library technology reports No. 51(3)) (p. 34).

Yelton, A. (2013). On Wilkinson, L. *is coding an essential library skill?* [Web log comment]. Retrieved December 14, 2015 from https://senseandreference.wordpress.com/2013/03/08/is-coding-an-essential-library-skill/

Yelton, A. (2012). *why should librarians learn python? (a better answer)*. Retrieved from http://andromedayelton.com/blog/2012/08/28/why-should-librarians-learn-python-a-better-answer/

Zou, Q. (2015). A novel open source approach to monitor ezproxy users' activities. *The Code4Lib Journal*, (29). Retrieved from http://journal.code4lib.org/articles/10589