PARTNERSHIP
The Canadian Journal of Library and Information Practice and Research
Revue canadienne de la pratique et de la recherche en bibliothéconomie et sciences de l'information

vol. 12, no. 1 (2017)
DOI: http://dx.doi.org/10.21083/partnership.v12i1.3961

Coding and Professional Development—Part 1: A Study in Contradictions

Sam Popowich
Discovery and Web Services Librarian
University of Alberta
Sam.Popowich@ualberta.ca

This article is Part 1 of a two-part article. Please see Part 2.

Abstract

This article is a critical examination of discourses around the democratization of software development skills. It critiques the idea of a "pipeline problem" in software development, while offering a theoretical perspective on class and gender bias in the adoption of programming skills.

Keywords

critical theory; information technology; software development; class; gender

Introduction

The rapid rate of technological change initiated by the industrial revolution in the 18th century has given technology—and technological skill—an ambiguity in the context of 21st-century capitalism.

Many tout the democratization of software development skills brought about by open-source tools and projects in the early 1990s as evidence of the possibility of a “sharing economy” free of the constraints of the capitalist mode of production (DiBona, Ockman, and Stone, 1999). Others, like David Golumbia (2009), dispute the emancipatory potential of the “digital renaissance”, arguing that “computationalism often serves the ends of entrenched power despite being framed in terms of distributed power and democratic participation” (p. 4).

The ambiguity between the dystopic and emancipatory power of technology plays out in many ways, not least in the realm of professional development. Here, the acquisition of skills can be viewed naïvely as a way to advance one's career. On the other hand, the skills themselves implicate workers more and more in a logic of automated technology that threatens to overwhelm them.

In addition, the "pipeline problem," first posited around 2012, framed the problem of the underrepresentation of women in technology as a deficit in skills acquisition. The *New York Times* argued that throughout secondary and post-secondary education "the pipeline" of skilled workers

> becomes increasingly porous, losing women at a discouraging rate…. On average, about 18 percent of computer science degrees go to women, and the Bureau of Labor Statistics reports that 19 percent of software developers are women. But experts say that at many prominent tech firms, where coding is king, the percentage of female programmers is in the single digits. (Hafner, 2012)

The solution to the pipeline problem was initially seen as a problem with supply—how to keep girls interested in science and technology in order to expand the pipeline into technology jobs. Framed as a deficit in education and interest, the pipeline problem ignores structural and institutional dynamics of both class- and gender-based power:

> If we want to solve the pipeline problem, the first step is telling young women what computer science is. *Computer scientist* needs to be a clear career path that little girls can envision and aspire to, the way they do with careers in medicine and law. Let's include computer scientist protagonists in children's books and create toys that allow kids to "play programmer" as easily as they "play doctor." (Jong, 2012)

More recently, however, the "pipeline problem" has been criticized for its naïve and parochial view of women's place in technology work. Rachel Thomas has argued that "because of the high attrition rate for women working in tech, teaching more girls and women to code is not enough to solve this problem" (2015) precisely because of gender bias in hiring and sexist climates at work. Writing in TechCrunch, Swati Mylavarapu (2016) writes that:

> Aptitude isn't the deterrent; it's culture. Dude-dominated images of programmers from popular culture, the lack of female role models, and broader societal attitudes towards women in tech make careers in our industry unappealing to many young women.

Miriam Posner (2017) further argues that even *when* women are working in technology fields, they face a gendered division of labour that devalues aspects of technology work seen as particularly feminized:

> The technology industry enforces a distinct gender hierarchy between front-end and back-end development. Women are typecast as front-end developers, while men work on the back end—where they generally earn significantly more money

> than their front-end counterparts. That's not to say that women only work on the front end, or that men only work on the back end—far from it. But developers tell me that the stereotype is real.

From a Marxist perspective, the socio-economic causes and effects of the gender problem in technology are part of the broader problem of technology in capitalism. Professional development in programming can be seen as both emancipating and oppressive: it ameliorates the pipeline problem but also reproduces structures of gender- and class-based power. In the same way, technology as a whole can be seen as both supporting the domination of the capitalist class over workers *and* as providing a site of potential resistance to that domination. Both of these processes operate through the dynamic of increased automation. For Marxists, it is precisely these ambiguities, these contradictions, which give workers the space to escape from the dominant logic of capitalist exploitation.

In the library world, the past five years have seen an explosion of interest in computer programming among library staff and initiatives to support acquiring the necessary skills. Workshops like Ladies Learning Code (itself an attempt to address the pipeline problem) and Software Carpentry are hugely popular, as are online tutorial systems like CodeSchool and CodeAcademy, and programming MOOCs from Coursera and EdX.

Given libraries' history and relationship to technology, skill-development initiatives are complicated both by the same factors present in the broader technology industry and by their own particular discourses. In some cases, library administrators and staff adopt a techno-utopian view of the world, as evidenced by the Digital Library Federation's (DLF) mission statement:

> The Digital Library Federation is a robust and ever more diverse and inclusive community of practitioners who advance research, learning, social justice, and the public good through the creative design and wise application of digital library technologies. (DLF, n.d.)

On the other hand, library workers sometimes see technology only as a modern tool for carrying out traditional library work. Both of these points of view can provoke resistance—both progressive and conservative—at all levels of the library hierarchy. Librarianship is both a highly technological profession and a profession with a well-developed suspicion of new technologies, perhaps epitomized by Michael Gorman's (1998) dictum: "I will embrace library technology, but not blindly" (p. 98).

Cultural Logic

In *The Cultural Logic of Computation* (2009), David Golumbia writes that technology cannot be extricated from the cultural contexts in which it is embedded and that the goal of a cultural investigation of technology is not to understand technology, but to gain insights into our own culture. For Golumbia, "the rhetoric of computation, and the belief-system associated with it, benefits and fits into established structures of institutional power" (p. 3). As a result, programmers occupy an ambiguous space in 21st-century

capitalism. In many ways, they are a privileged elite consumed with a spurious ideology of meritocracy and techno-libertarianism.

On the other hand, many programmers hold jobs of extreme precarity and endure racial and sexual discrimination and harassment. Nick Dyer-Witheford (2015) describes this "programming proletariat" as having the trappings of privileged, emancipated, creative work which only disguises their proletarian identity:

> The process by which "semi-autonomous experts" and other professionals win a limited independence from capital on the basis of special technological knowledge, only to find their autonomy and bargaining power whittled down by the very automating and networking dynamics they helped set in motion is an important part of "re-proletarianization." (p. 66)

Indeed, this process applies to librarians as well—hitherto a relatively privileged sector of the working class, being drawn more and more into the logic of the "capitalist university" (Heller, 2016) on the one hand and precarious public-sector work (Groover, 2014) on the other.

From a Marxist perspective, technology is never wholly on one side or the other of a power dynamic; Marxism sees the dynamic itself as a dialectical whole in which both "entrenched power" and "democracy" are moments in a process that must be transcended. When librarian researchers like Karen Coyle (2016) state that "libraries of course have been technology-based from the beginning of their history" (p. 48), they don't go far enough. Librarianship is a profession that both relies upon and resists technology in certain ways, and libraries have always been situated at a nexus of institutional power (from monasteries to Andrew Carnegie) and information technologies (from the scroll to the ebook). We must be aware of how technology both serves powerful interests, ensuring for example that women remain a marginal, exploited group within both the technology and library technology fields, *and* offers a means by which such power might be captured and redistributed to create more equitable and participatory culture.

Prior to the 1980s, librarianship's use of technology went more or less unremarked. It is only now, since the rapid development of the web and open-source software, that our reliance on technology seems somehow new, strange, a disruption. The contradiction between a "traditional" view of librarianship and one that sees technologies of software as essential to library work is only one tension in the totality of the technological discourses within the profession. This article will use Marx's theory of technology as a way to look at these discourses, with an aim to showing a way forward that might transcend these tensions and contradictions.

The Changing State of Library Technology

It's possible that—pre-automation, pre-web—library workers might have been able to come out of library school with a well-defined set of skills (cataloguing rules, reference interactions, etc.) and go through their entire career without needing to upgrade or "re-

skill". The 2nd edition of the Anglo-American Cataloguing Rules (AACR2) was the dominant cataloguing standard from 1978 until the release of RDA in 2010. Michael Gorman, one of the authors of AACR2, was vehemently opposed to the introduction of RDA, writing that revisions to AACR were being "hijacked" by those who valued theory over practice, and that "the practical result of their theoretical approach promises to be the biggest disaster to hit descriptive cataloguing since the draft rules of 1941" (Gorman, 2007). Cataloguers have long memories, perhaps both as a cause and an effect of the slow pace of change in librarianship as a profession.

At least until recently. Since the mid-1990s, the pace of change in library technology has sped up, driven by two big shifts: automation in the late 1970s and early '80s, and the post-dot-com web of the late 1990s. However, the effect has been uneven; in many ways we continue to think about services and workflows in a pre-digital, pre-web way, which contributes not only to the divide between digital services and more traditional library work, but also to a misunderstanding of the critical importance of maintaining skills in a post-AACR2, automated, digital world.

Michael Schofield (2016), an important voice in the User Experience field, argues that

> [b]oth the relevance and agency of libraries pivot around technical knowledge of the people at its core. Our understanding and mastery of the web is what gives voice to the base values of librarianship: Safiya Noble bringing to light the politics inherent in algorithms turns the lens inward on that thing at the crux of what libraries do in part because that thing is already or will be a computer program and thus some librarians must become programmers to articulate and continue the mission.

Libraries that recognize this must figure out a way to financially support ongoing professional development beyond the occasional conference. To my mind, the fact that this has to be argued in 2016 at all is an indication that many libraries still do not recognize the benefit of fundamental, low-level software development skills for library workers in many areas, such as the automation of routing cataloguing and metadata tasks or the manipulation of spreadsheet data in all units of the library. Like the initial approach to the pipeline problem, many libraries see the acquisition of these skills as something that should happen elsewhere (in the pipeline); others simply think that those skills are unnecessary for anyone not working in a (predominantly male) library technology or IT support unit.

Librarians who work in IT or digital service units *may* be able to carve out time for self-study or find the time and funding for small-scale workshops like Ladies Learning Code (1 day) or Software Carpentry (2 days). But even this is often impossible for library workers who are not recognized as engaging with the digital on a daily basis. Inequity of workload—especially among junior staff, and most especially junior female staff—contributes to an inequity of opportunity for professional development.

The acquisition of new skills, whether for promotion/position change or simply to stay abreast of new developments, is often difficult for IT librarians and nearly impossible for

others. How many web librarians are sufficiently conversant with the evolution of JavaScript to engage with Schofield's (2016) statement that "the seam-bursting popularity of React calls into question the pros of writing the web in anything but JavaScript"? How many of them have overcome the imposter syndrome fostered by reading something like that in order to learn all the implied skills on their own?

In non-IT/digital units, the ubiquity of digital technology provides both opportunities and challenges. Safiya Noble's work on algorithmic bias is a prime example of how a deep understanding of software can contribute to information literacy and source evaluation. Perhaps a librarian with a deep understanding of web software and search algorithms might have prevented the former British MP Louise Mensch from mistaking her own search history for autosuggested Twitter results (Rawlinson, 2015).

Librarians Learning Code

In 2015, *Library Technology Reports* (*LTS*) dedicated an entire issue to "Coding for Librarians: Learning by Example" (Yelton, 2015). The purpose of her survey and report, Yelton writes, was to answer the question of how librarians use code in their daily work. The impetus behind this study was the rise of programming tutorials and hackathons not only in tech-focused arenas (such as the Access and Code4Lib conferences), but in more general-purpose library conferences as well. Yelton writes that "these short workshops are wonderful for exposing people to fundamental concepts and creating positive experiences around code, but students don't necessarily know what to do next" (p. 5).

This issue of *LTS* provides an excellent starting point for any library worker interested in learning to code; Yelton provides links to resources and helpful commentary from interviewees and the respondents to her survey. But she also points out that "while managers who code understand uniformly the value in supporting this skill, not everyone is lucky enough to have such a manager." Some librarians "have no support, or even face active hostility" while some institutions "simply don't have pertinent checkboxes in their paperwork, and it's hard to [explain] the relevance of these skills to a faceless bureaucracy" (p. 29).

One of the contradictions, then, in continuing professional development in coding is the tension between views of "traditional" library work that do not include the digital (at least, not at the level of actual programming) and the recognition that the library now requires the ability to interact with our digital infrastructure at a low level.

One of the reasons for this tension is that we are slow to recognize changes in our conditions of work, especially the transition from material to immaterial labour. While librarianship has always been intellectual work, for much of its history it was an extremely material practice: the control and movement of physical books, the production of physical catalogues (both book and card catalogues). Indeed, the decline of "library hand" in the late 1960s and early '70s (Morton, 2017) corresponds with the invention and spread of Machine-Readable Cataloguing (MARC), invented by Henriette Avram between 1966 and 1968 (Avram, 1975, p. 5-8). The transition from material to

immaterial labour occurred as part of a larger transition within capitalism that saw, among other things, the rise of the neoliberal state and new geopolitical order, as well as the transition from societies of physical discipline to societies of (technologically enabled) control (Hardt & Negri, 2000, pp. 23-30).

Library work in 2017 is immaterial, dealing among other things with electronic records management, databases, software applications, electronic communications, and hacking (writing scripts for particular purposes, sometimes only needed for a single run). Yelton quotes a respondent whose MARC template script had been successfully modified (hacked) to produce RDA by a cataloguing staff member (p. 27).

In addition to hacking, library work also increasingly involves major, structured software development projects coming out of the open-source ILS world (Evergreen, Koha) and taking off with the advent of open-source discovery systems (Blacklight, VuFind) and digital asset management systems (Hydra, Islandora). More and more, “hacking” coexists with full-scale software projects in a continuum of rigour, ephemerality, and best-practice adoption.

But the distinction is also a site of discrimination based on both gender and class (Cowles, 2017). Within the University of Alberta Libraries, we work on both small-scale hacking projects, such as tabulating page numbers for digitation projects, or building reports of single-subscription ejournals, with large-scale, multi-developer projects like the Hydra-based Education & Research Archive (ERA). In the second part of this article, I will look at the ways in which organizational culture affect technology skills development, particularly in computer programming.

The hacking vs. programming tension is, in my opinion, a productive one—as long as staff members can insert themselves according to their comfort level. This enables them to regain some agency in the face of often top-down technology decisions and learn through working together both internally and with members of larger open-source communities. Even here, however, professional development opportunities for “developers” are no more forthcoming than for “hackers” if an organization does not recognize the centrality of programming to (increasingly immaterial) library work.

This contradiction, between low-level digital work being central to library labour and the inadequate recognition and support among library administrators, can provide an opportunity from bottom-up communal professional development, but only with the expense of political capital, which raises all sorts of issues around privilege and power within our institutions.

References

Avram, H. D. (1975). *MARC: Its history and implications.* Washington, DC: Library of Congress.

Cowles, E. (2017, March 10). not a developer. *Escowles.github.io.*

Coyle, K. (2016). *FRBR, before and after: A look at our bibliographic models.* Chicago, IL: ALA.

DiBona, C., Ockman, S., & Stone, M. (1999). *Open Sources: Voices from the open source revolution.* Sebastopol, CA: O'Reilly Media.

DLF. About the Digital Library Federation. *DLF: Digital Library Federation.*

Dyer-Witheford, N. (2015). *Cyber-proletariat: Global labour in the digital vortex.* Toronto, ON: Between the Lines.

Golumbia, D. (2009) *The Cultural Logic of Computation.* (2009). Cambridge, MA: Harvard University Press.

Gorman, M. (1998). *Our singular strengths: Meditations for librarians.* Chicago, IL: ALA.

Gorman, M. ([ca. 2007]). RDA: The coming cataloguing debacle. *Special Libraries Cataloguing.*

Groover, M. (2014, January 6). On Precarity. *Bibliocracy*.

Hafner, K. (2012, April 12). 'Pipeline' to programming jobs has leaks. *The New York Times*.

Hardt, M., & Negri, A. (2000). *Empire*. Cambridge, MA: Harvard University Press.

Heller, H. (2016). *The capitalist university: The transformations of higher education in the United States, 1945-2016.* London, England: Pluto Press.

Jong, A. (2012, March 15). Solving the pipeline problem: How to get more women in tech. *Forbes*.

Morton, A. (2017, February 16). Library hand, the fastidiously neat penmanship style made for card catalogs. *Atlas Obscura*.

Mylavarapu, S. (2016, May 10). The lack of women in tech is more than a pipeline problem. *TechCrunch*.

Posner, M. (2017, March 14). We can teach women to code, but that just creates another problem. *The Guardian*.

Rawlinson, K. (2015, August 22). Louise Mensch takes Twitter swipe at Corbyn campaign—and hits herself. *The Guardian.*

Schofield, M. (2016, May 9). Web education must go further than a conference budget. *LibUX*.

Thomas, R. (2015, July 27). If you think women in tech is just a pipeline problem, you haven't been paying attention. *Medium*.

Yelton, A. (2015). Coding for Librarians: Learning by Example. *Library Technology Reports, 51*(3).