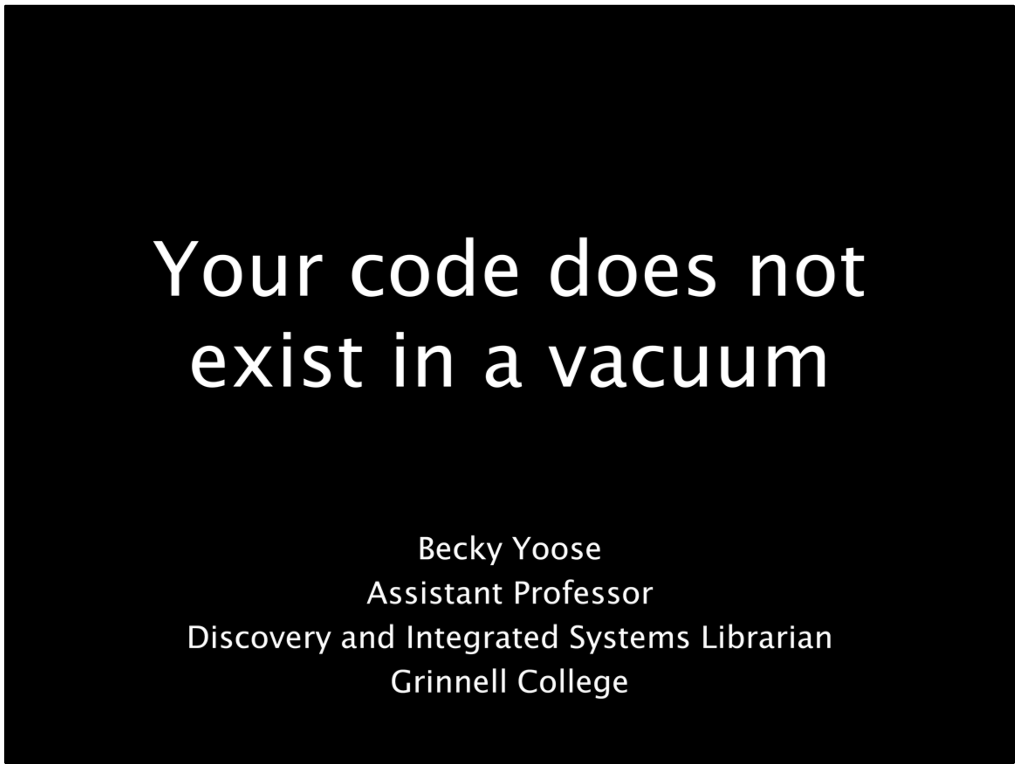
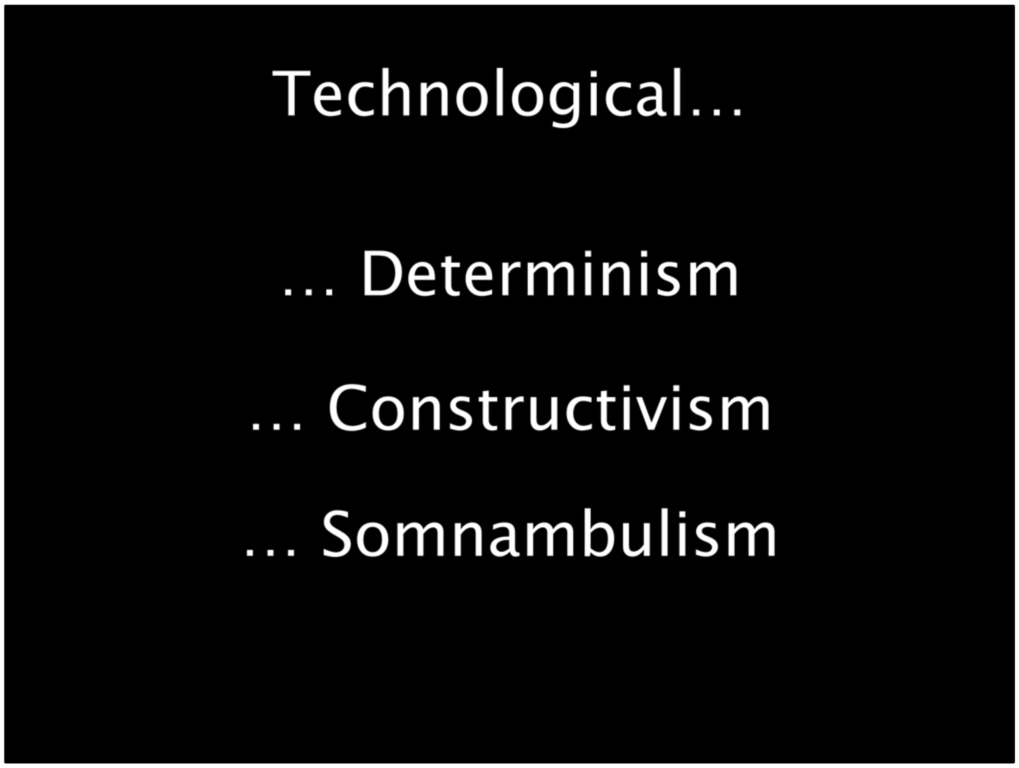


Your code does not exist in a vacuum

Becky Yoose
Assistant Professor
Discovery and Integrated Systems Librarian
Grinnell College

Good morning everyone. Today I give you a talk about technology. Actually, it's a specific program; the programming language I used is the American fork of English. The program was written in and optimized for the United States operating system. So, the program I'm about to execute has its limitations in other operating systems. YMMV.

When technologists talk technology, we tend to focus on certain aspects. We also have a collective unspoken agreement what the term "technology" includes. For example, the mention of language and nation-states as forms of technology might have caused a few of you to pause. If I would have said that the language I used was PHP, there would have been no pause. Technology is more than just lines of code. It is part of our lives and livelihoods in the case of library technology, or libtech, and it interacts with everything around us. The talk today is only a starting point in a discussion about technology, society, and libtech.



Technological…

… Determinism

… Constructivism

… Somnambulism

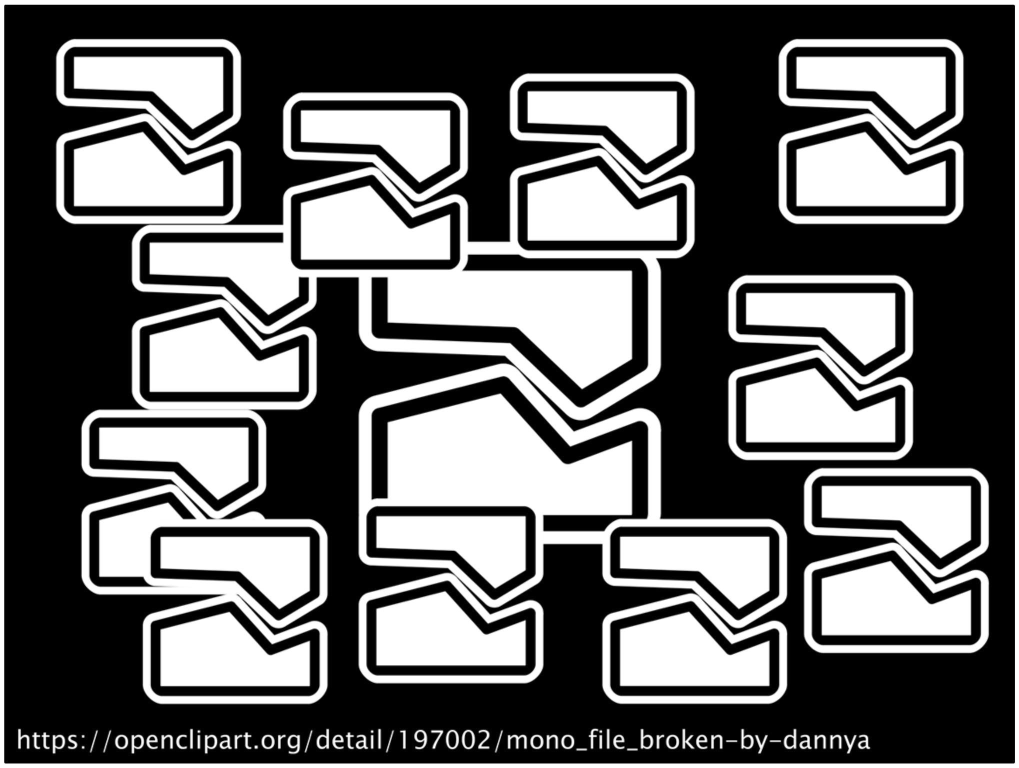
So when we talk about technology and society, there is a choice that we can make with the theoretical framework. One choice is the technology shaping society (or Technological Determinism). An example of this is some Print Culture scholars' argument that the moveable type printing press in Western Europe acted as an "Agent of Change" in terms of the formation of modern Christianity via the Reformation.

Another choice has society shaping technology (Technological Constructivism). Programmers might be familiar with an example of this in the way of Conway's Law: a system's design is dictated by the organization's communication structures.

These choices are not mutually exclusive, though. There is still another choice, one that focuses on the current state of technology and society - Technological Somnambulism, defined by political theorist Langdon Winner. In his essay "Technology as forms of life," Winner argues that we do not fully understand technology's relationship with society. We view technology as a tool to be used, without considering its long term implications. Society fails to recognize the way that technology essentially creates new worlds for society to operate in. In essence, we are "sleepwalking" with regard to our knowledge about how both technology and society affect one another.

With the following theoretical frameworks in mind, I will address the potential reshaping of the libtech world by certain tools and practices adopted by libtech. We tend to talk about software or hardware when we talk technology. However, I wish to focus on the production tools and practices as they themselves are forms of technology.

I will talk about two, the first being the role of failure in libtech.


https://openclipart.org/detail/197002/mono_file_broken-by-dannya

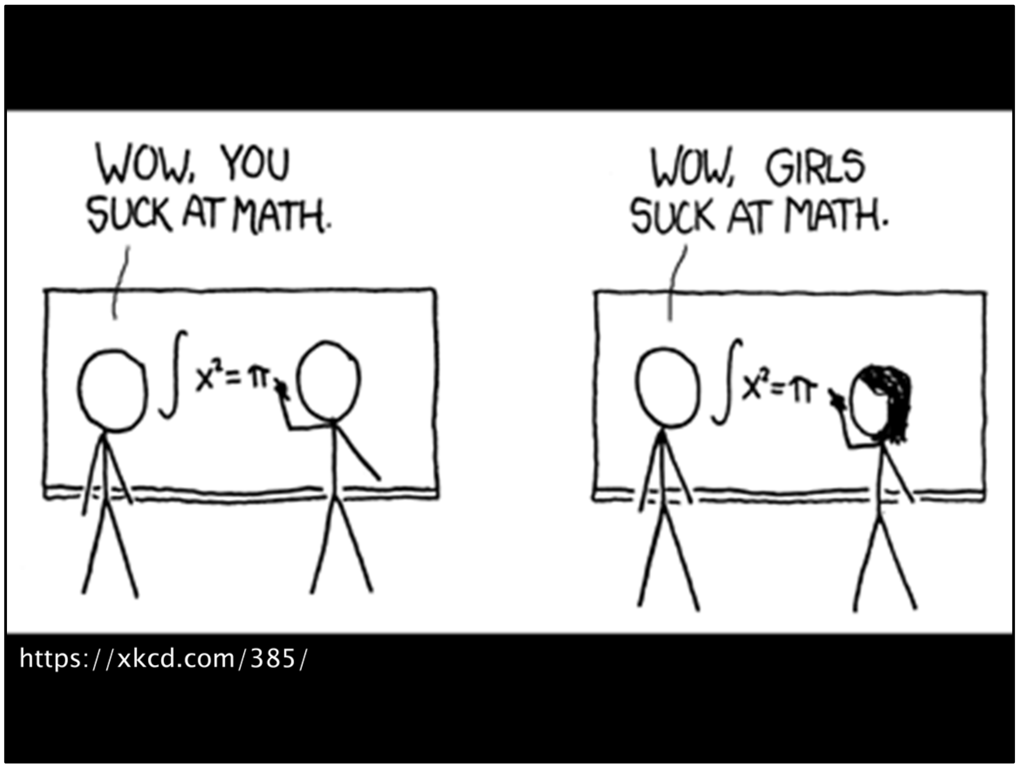
The “fail fast, fail often” mantra and failure acceptance has been making its way into libtech. An example is the Fail4lib preconference that has been running three years straight at code4lib. People come to share their failures in their technology projects: what caused the project to fail and lessons learned.

The failure mantra has its close relatives. Eric Raymond’ s “The Cathedral and the Bazaar” outlined “Release early, release often.” This pattern of developing code and releasing it into production in rapid increments has led to the popular Agile development method. Since its birth in the 2000s, Agile has become one of the most talked about, if not one of the most admired, software development methods in the last decade. The fast pace is geared towards building a product that will meet user needs without getting stuck in the pitfalls of traditional development methods.

The fast failure mantra we see from startup culture makes several assumptions. First, you have resources to handle multiple failures in a short period of time. Second, you have resources to fail in rapid succession for long periods of time. Third, if one project doesn’t work out in the end, you can move on to another project until you get something that sticks. The fluidness of the technology startup scene is apparent in many tech workers’ resumes, jumping from startup to startup to startup, many of which do not last a couple of years in existence.

Failure has monetary, intrapersonal, and interpersonal costs. If one is not wise in deciding when failure is an acceptable risk, they not only risk straining already limited budgets, but also damage to staff morale, stakeholder relationships, and user trust. If you keep releasing new products, only to have them squander in beta or abandon them for a newer project, what does that say to your users? In addition, project hopping in rapid succession leads to burnout risks. In fact, one of the main documented weaknesses of Agile is staff burnout. The relationship of technology and failure also has an effect on library administration, who face both internal and external pressures to ensure that their library operates efficiently while providing relevant user services. Doing failure for the sake of failure is a costly venture, and one that many administrations and budgets cannot support long term.

Now, there are those who are successful in adopting the fast failure mantra in libtech…



https://xkcd.com/385/

Which leads us to the fourth assumption of the failure mantra: having the privilege to fail without catastrophic consequences to one's professional and personal lives.

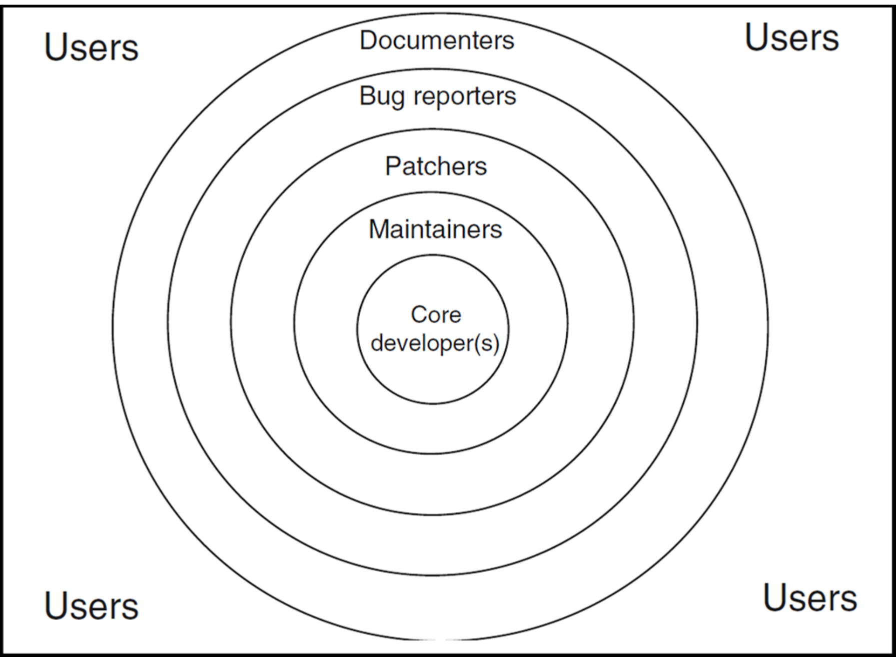
In most failure narratives in the United States, you have young white businessmen and male scientists enduring failure after failure until they achieve success. In these failure narratives, failure is venerated almost as a rite of passage, and that, with each failure, the person develops and nurtures grit and perseverance that ultimately lead to his success.

Similar narratives for anyone outside of the young white man mold are uncommon at best. Those who do not fit the mold are subject to closer scrutiny of their actions and judgments. Any mistake or failure then becomes magnified and given more weight for consideration not only for that person, but for "The Other" that they represent. Therefore, obligatory XKCD. The "glass cliff" that exists in senior leadership for underrepresented populations, I argue, exists for libtech, where "The Other" are put in positions where failure is almost inevitable due to circumstances mostly outside of their control.

Yes, librarianship is made up of a majority of white women, but the failure narrative does not change to reflect this majority. There are additional gender considerations regarding librarianship and technology in this aspect. There is not enough time in this talk to go through all the gender and racial status, power dynamics, and their implications within librarianship in the past century; however, when a profession that has been assigned characteristics of one gender is combined with tools with conflicting gender assignments, the gender characteristics assigned to that profession does not change based on the tools they use. What does change, however, is the type of person in the profession who handles the tools. This is evident when one looks at the gender balance of libtech. In addition the power dynamics within the relationship of technology and librarianship can be readily seen with commercial libtech vendors, and the object failure on the library world's part to hold them accountable to provide affordable technology that meets the needs of both library staff and users. Instead, these vendors invoke a paternalistic role in their relationship with libraries, who are limited in socially acceptable actions for disagreement or assertiveness in the relationship. If you can't get what you want from a vendor, but at the same time can't afford the risk involved with the fail fast mantra, what options are left?

Failure, as it is used in development, is a double-edge sword. Without failure, code cannot evolve and thrive. The focus on adaptiveness and responsiveness in the failure mantra gives the library the means to provide forward looking services to the world. However, for those who do not have the resources or privilege required for "fail fast, fail often" the cost is greater than what most realize.

[Sidenote - LITA 2014 keynote speaker AnnMarie Thomas touched on failure as a privilege; however, she then went on to say that failure should be rebranded as iterations – essentially adopting an Agile development terminology. Rebranding only goes so far in disassociating one term's meaning from another. See also "lipstick on a pig."]



Ducheneaut, Nicolas. 2005. "Socialization in an Open Source Software Community: A Socio-Technical Analysis." *Computer Supported Cooperative Work: CSCW: An International Journal* 14: 323-68. Using figure under Fair Use.

The second production facet in this talk is one that hits close to home for libtech.

The open source community provides a place where we, libraries, archives, and museums, can develop software to aid in the creation, maintenance, and providing access to information. While libtech sees value in collaboration and sharing library software and technology, we find ourselves dealing with incorporating a product of certain socioeconomical beliefs.

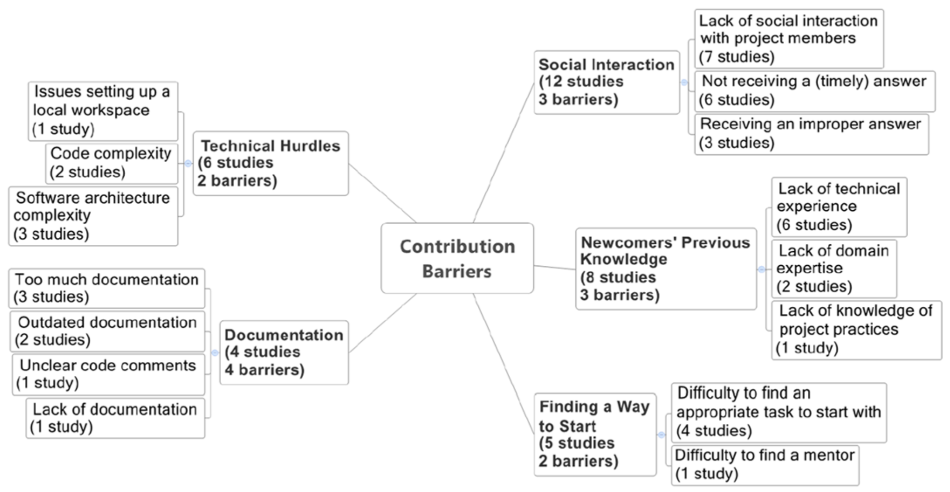
There are two views of the general OS community that are in play here. The first view is one of collaboration: people coming together to share their skills and talents to create something, share this something, and hope that others will build on it. Collaboration and sharing seem to be in alignment with many of our organizations' missions.

The second view of the OS community is one of meritocracy. One's skills and work ethic are essential in the value they bring into the community; therefore, determining status within that community.

Meritocracy is a very dangerous word to throw around without a qualifier in this context. We know through various observations that meritocracy refers to a certain subset of the greater OSS community – the majority race, gender, sexuality, and social class within technology – but even within this limited scope, not all contributions are considered equal. In the figure on the slide you see one researcher's organization of a OSS project community; the bull's-eye is where code developers lives, followed by the maintainers, patchers, bug reporters, documenters, and outside you have the users. Another researcher refers to the structural hierarchy of OS projects as an “onion patch” of sorts.

This figure also follows the social hierarchy of OS, with those creating substantial code, or deemed to have superior coding knowledge and skills, holding the highest status. Those with lesser coding skills or have less time to contribute code back into the base are set lower in the social hierarchy. There are those in OS who advocate that non-code contributions are as important as committing code, but the general social hierarchy has not changed in large part to reflect this sentiment. I see this in libtech listservs: people want to use OS libtech software but are very apologetic in not being able to contribute code, trying not to appear as what some in the greater OS community publically call “free loaders,” even “leeches.” Even at a technology conference where the community says they value non-code contributions, there is still a tinge of shame felt by certain folks for not being able to contribute code back to their peers.

The inherent social status gained with coding skills also reinforces the class structure within librarianship. The greater your coding knowledge, the more weight you have in the library community. Coding and technology skills are highly sought after by those in the lower classes of librarianship to gain status in their workplace and their field. This is why learning to code workshops are packed; if one has the skill, they benefit from the boost they get in the social and professional networks. For us catalogers and metadata folks, the push to learn coding in our circles has been relentless for years. Learning to code can be empowering for folks who otherwise have no other options available to them, and yet we end up using our coding skills as a way to try to move up the class system in librarianship, to gain more respect from our peers, and to gain a spot at the table where the important decisions are made.



Contribution Barriers
Technical Hurdles (6 studies 2 barriers)
Issues setting up a local workspace (1 study)
Code complexity (2 studies)
Software architecture complexity (3 studies)
Documentation (4 studies 4 barriers)
Too much documentation (3 studies)
Outdated documentation (2 studies)
Unclear code comments (1 study)
Lack of documentation (1 study)
Social Interaction (12 studies 3 barriers)
Lack of social interaction with project members (7 studies)
Not receiving a (timely) answer (6 studies)
Receiving an improper answer (3 studies)
Newcomers' Previous Knowledge (8 studies 3 barriers)
Lack of technical experience (6 studies)
Lack of domain expertise (2 studies)
Lack of knowledge of project practices (1 study)
Finding a Way to Start (5 studies 2 barriers)
Difficulty to find an appropriate task to start with (4 studies)
Difficulty to find a mentor (1 study)

Steinmacher, Igor, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F. Redmiles. "A Systematic Literature Review on the Barriers Faced by Newcomers to Open Source Software Projects." *Information and Software Technology* 59 (March 2015): 67–85. Using figure under Fair Use.
Figure from pre-print at http://flosscoach.com/publications/IST2015.pdf

Open source development assumes that you also have resources to make useful contributions to the software. Some of the biggest OSS projects in libtech list major institutions, organizations, and vendors in their ranks, and they drive major development and maintenance of the project, mirroring some of the biggest general OSS projects.

When your OSS project uses a set of tools, you are making the assumption that the users of your project will have access to the same tools. In addition, you are assuming that they have the skills and resources to install and maintain the stack that comes with your project. Even though you might think that your project would benefit the community, the reality is that libtech OSS comes with a resource cost for those organizations that do not have similar access to skills and resources. This cost is sometimes paid by kludging together the software, but kludges only get you so far, and are prohibitively expensive to maintain long-term.

If you do not have the resources to effectively install and maintain OSS at your workplace, where do you go? There are vendors that support libtech OSS, but then you are stuck in the dilemma of using OSS but not really "owning" your installation. This has implications with social status within the community; the failure of not having the skills to run OSS with limited resources. Again, we find ourselves back to narratives of success in the face of extremely limited resources. If you want it bad enough and if you have the skill and grit to get through the failures, you will succeed. If you can't make it work, well… This does not even scratch the surface with the problems associated with newcomers wanting to contribute to OSS projects. Again, I have a nice figure for you all to look at from a recent study of the literature regarding barriers to OSS involvement for newcomers, ranging from documentation and technical knowledge to social interactions.

On the flip side, you have the problem faced the owners of the OSS project: creating the code, maintaining it, expanding and refining, and so on. The labor costs associated with the maintenance of an OSS project places further constraints on resources for both the workplace and the individuals involved. Unless your project can work in vastly different environments with different resource levels, you WILL get inedited by questions from users. Depending on the perceived usefulness of your project, you might find yourself spending more resources on the project than first anticipated. If your workplace decides that you shouldn’t spend more time on your project, what do you do? Do you try to recruit maintainers and devs who are already working on other projects to get to that critical mass where the project would be self sustaining? Failing that, do you stop development? Or do you maintain it on your own time? If you choose personal time, what does that do to workplace expectations? What has to give in order to keep up the project?

In short, open source community and production structures have their costs and implications in libtech, even when we share and collaborate with the best intentions in mind.

Where do we go from here?

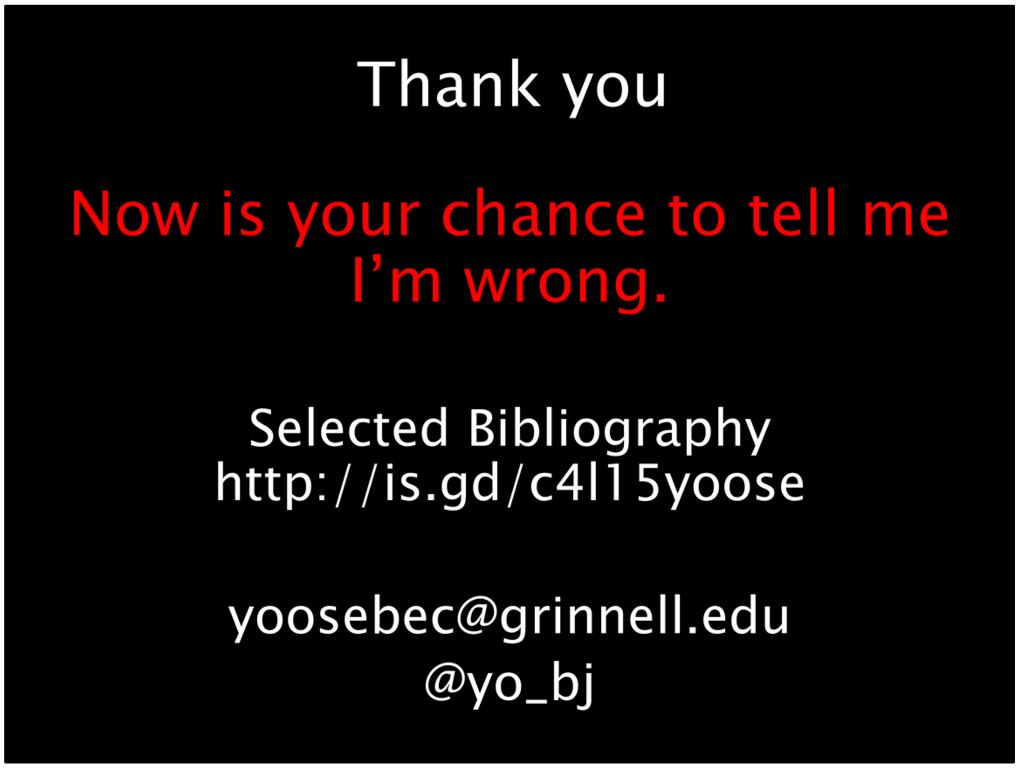
So, where do we go from here?

This is the part where I ask the really messy questions: Why are we adopting these tools and ideologies in libtech? Why do we think that they are best suited for the libtech and GLAM environments? Is it to emulate the success that others have had, or to bring in that desperately needed feeling of relevance to our place in the world?

Coming back to the effects of these tools and practices in libtech, we find ourselves in a precarious position. Let me be clear: I am not saying that we should be risk adverse. Failure is needed to grow. I am not saying that we should not collaborate openly in software development and maintenance. The sharing and collaboration in open source speaks to our institutions on an altruistic level. And yet the tools that libtech reaches for come with baggage that we haven't fully inspected, thus putting us back to the state of technological somnambulism.

This is a dangerous place to be, both practically and philosophically.

It's time to start inspecting the luggage. The things that we reach for are not actually tools, but potential worlds for libtech to operate in. I wish I can give you a neatly packaged checklist to use in inspecting, but there is none, and I'm not sure if we can all agree on said checklist. I can, however, suggest a starting question for you to ask yourselves. Winner asks "As we 'make things work,' what kind of *world* are we making?" As we develop, hack, and tweak libtech – both software and community – what exactly is the world that we are making, and does it match the world we *want* to make?


Thank you

Now is your chance to tell me I'm wrong.

Selected Bibliography
http://is.gd/c4l15yoose

yoosebec@grinnell.edu
@yo_bj

[Side note: +1 to Julia Bauder, @ranti, @kayiwa, and others for helping with the refinement of the presentation.]