Good morning, my name is Becky Yoose, and I’m here to tell you a story.
Before I tell the story, however, please note the following disclaimers.

Disclaimers

Names will be named

Bad words will be said

Cats will be memed
This tale starts off with a group with a need not met by existing repository software. The group decides that their need would be best met with a homegrown system. They have several developers to build the system, as well as access to resources for acquiring the necessary infrastructure and training needed to complete the project.

The group releases their code as open source, and they lived happily ever after.
Now let us look at another group with a need not met by existing commercial software. They too have materials of cultural, scholarly, and historical importance.

The group decides to go with the open source system that the first group developed.

The second group, however, does not have a dedicated developer on staff, but are dependent on one or two “part-time” technical staff. The budget is very limited, including money for training and resources.

They start the project, but find themselves facing roadblocks in installation and basic configuration. Documentation, if they could find it in the first place, was vague, outdated, or went over the heads of the staff reading it.

They decide to call in a vendor that supports the system for help. No answer. After several more attempts, the vendor answers, only to reply that they are overwhelmed by other groups asking for help.

https://secure.flickr.com/photos/miz_curse_10/1404420256/
The group gets the repository up and running, but only precariously. The group feels like they are only taking, and not giving back to the community because of their frequent pleas for help and lack of resources.

This is where the story ends for now.

The story I just told was a conglomeration of the experiences that many small organizations have faced when trying to enter and participate in the open source repository community. Grinnell College implemented Islandora 6 in 2012, and it was a very bumpy ride. But because we went through the process with an OS repository, other small organizations contacted us to share their repo experiences and ask questions. I learned in this process that we – the small and limited resourced – all shared similar experiences.

https://secure.flickr.com/photos/jollino/5588370858
https://secure.flickr.com/photos/nytisn_pix/386214544
In 2013, Grinnell joined four other small liberal arts colleges using Islandora to form the Islandora Consortial Group.

The group pools together different skill sets to help other group members with their repositories; for example, one person in the group has experience with systems administration, while another person has experience in Drupal module development. Together we are working on extending a version of Islandora 7 for ICG members to base future development work on, as well as ensuring that common core needs of the group are met.

However, the name of the group is a bit misleading now. We have another SLAC who is participating, but they’re a Hydra shop.
In short, we have two communities: those who have the resources to develop and shape both the software and the community, and those who are struggling on their own, and pooling resources with other groups just to make their repositories functional.

What can we do to bridge these communities?
The goto solution from the community has been training. However, note the red asterisk.

Training is expensive, with one or two events usually costing more than the annual staff training budget. Because of training cost, only one or two people are trained. In addition, some groups can’t afford to provide market competitive compensation for certain skillsets, be it in salary or job rank. This combination creates a risk for the group. How? Well, what happens when that trained person leaves the group?

Training should be approached as an option, not a requirement to use your software.
Seriously, what can we do?

So the current community approach toward training is only treating the symptom. How can we better integrate the community, then?

I don't have answers, but I have two suggestions.
The first is documentation. Don't just do it, do it right. Test your documentation like you test your software. Have people outside your team try and follow your docs. Fix what’s broken. Rinse, repeat.

If you need a place to get started, Write the Docs is a great resource for technical documentation tools and best practices.
Here is the second. What do I mean by echo chamber? If the majority of places using your software with the least amount of issues look like your group in terms of skill sets and resources, congrats – you have echo chambered your software.

If a group can’t get your software to work, they more than likely can’t participate in shaping the cultural memory and scholarly record. Think about that for a moment. The amount of resources spent on yak shaving so they can get a basic installation off the ground and stable enough to maintain means that there is no chance to go beyond treading water in the community, if they can tread the water in the first place.

Getting out of the echo chamber requires at least these steps, in addition to taking on a co-design development approach:

1. Look at your stack. What is the adoption rate for parts of that stack? What is the learning curve to implement it? I’m not saying that you can’t use newer languages or frameworks; I’m saying that if you know you’re using a stack that is not widely adopted by the greater community, you need to lower the bar considerably in order for the greater community to use your software.

2. Test your software in places that don’t look like yours. If they can’t implement your stack, fix it on your end. You have a responsibility to the community to make your software functional as the users in the community have a responsibility to send bug reports and to help troubleshoot.

Stop dev’ing OSS in a fucking echo chamber.

(Co-design your open source shit.)

http://is.gd/codesign
@ruebot @mjsuhonos @jordanheit:
Related: what I <3 about the Hydra architecture and philosophy is that we're not tied to Ruby, Fedora, etc.

-- @mjgiarlo, 20/2/2014

https://twitter.com/mjgiarlo/status/436546820227874816

I’ll end this rant with another story that comes to you in three tweets, starting with a tweet from a prominent member of the Hydra community. Note the comment about the Hydra architecture and its relation to Ruby.
@eosadler I got partway thru (after having to go thru hell&back to even FIND it), but then it was all "write Ruby code!" and I was all NO.

-- @LibSkrat, 21/4/2014

https://twitter.com/LibSkrat/status/458354454216310784

Tweet number two: a community member’s experience in trying to implement Hydra. Note her mention of Ruby.
Who did you build this system for? Because, let me tell you, none of the users can actually *use* it.

-- @satifice, 20/5/2014

https://twitter.com/satifice/status/468799592767946752

Tweet number three comes from a community member trying to implement an OS repository. Treat Nina’s question as non-rhetorical.
Let’s continue the discussion.

yoosebec@grinnell.edu
@yo bj
Image credits

• Slide 4:

• Slide 5:
  – https://secure.flickr.com/photos/jollino/5588370858
  – https://secure.flickr.com/photos/nuystsia_pix/386214544