

Building Scientific Software Communities

Daniel S. Katz
d.katz@ieee.org @danielskatz
Computation Institute, University of Chicago & Argonne National
Laboratory

(much content adapted from Open Source Summit 3: Ben Balter, Github & Jim Jagielski, Apache & Joseph Porcelli, GovDelivery & notes from http://ossummit.org/)





My Experience



- Working towards Sustainable Software for Science: Practice and Experiences (WSSSPE)
 - http://wssspe.researchcomputing.org.uk
 - https://github.com/danielskatz/WSSSPE
 - WSSSPE1 report
 - http://dx.doi.org/10.5334/jors.an (but...)
 - http://openresearchsoftware.metajnl.com/article/view/jors.an
- Application Skeletons
 - https://github.com/applicationskeleton/Skeleton
 - http://dx.doi.org/10.5281/zenodo.13750 (again but...)
- NSF http://www.nsf.gov/si2
- Open source is a not a panacea
 - Technology isn't difficult
 - But people can be





Building Open Source Communities



- Be committed & energetic leader/visionary
- Be clear about what you want
- Engage community
- Increase engagement
- Track engagement activities towards project goals



Be Clear About What You Want



- Treat an open source project like a political campaign
 - Paint a vision
 - Say where you want to go
 - Ask for help in getting there
- Requires ongoing involvement of team/community leaders
 - "Governance"





Governance & Communities



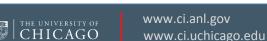
- Walled Garden
 - Open source through license only, no real community
 - Often run by corporation
 - Can have focused design, good usage
 - Apple app store as example



Governance & Communities



- Benevolent Dictatorship
 - One person has ultimate say
 - Ideally light touch, influence, final decisions
 - Mandate from the community (who can leave if not satisfied)
 - Linux kernel best known example, also common in languages
 - Question about long-term sustainability
 - Do new people want to join?
 - What happens when dictator leaves?



Governance & Communities



- Meritocracy
 - Flat layer of peers
 - Forces the community to work together
 - Provides a neutral place for individuals and companies to work together
 - Merit earned by deeds, not position or reputation
 - Example: all Apache projects

 Each project may have a "natural" governance model

Adapted from Jim Jagielski

www.ci.uchicago.edu

www.ci.anl.gov





Engage Community



- Why?
 - More hands -> quicker work
 - More minds -> better solutions
- Engagement: meaningful and valuable actions that produce a measurable result
- Engagement = Motivation + Support Friction
 - Intrinsic motivation: self-fulfillment, altruism, satisfaction, accomplishment, pleasure of sharing, curiosity, real contribution to science
 - Extrinsic motivation: job, rewards, recognition, influence, knowledge, relationships, community membership
 - Support: ease, relevance, timeliness, value
 - Friction: technology, time, access, knowledge

Reduce Friction: Project Setup



- Reduce Friction
 - Do: Use a familiar license
 - Do: Spell things out in the project readme
 - Do: Include a LICENSE.txt file
 - Do: Include a CONTRIBUTING.md file
 - Don't: Require a Contributor License Agreement
 - Do: explain context, relationship to other projects
 - What software/services do you think are available to use
 - What do you plan to build





Copyright



- Creators have exclusive rights, including
 - Reproduce (copies)
 - Create derivative works
 - Distribute copies
 - Perform/display publicly
 - Sell/assign rights to others (license)
 - Transmit

- Since 1979, copyright is automatic at time of creation (at least in US)
- Can't copyright ideas, only embodiments

Adapted from Ben Balter

Licensing



- Distinguishes rights & use from ownership
 - What can and can't I do with your code
 - If I contribute, what rights do I waive? (think of it like an apartment lease)
- Software Licensing
 - Describe what rights I'm granting to you
 - Disclaim that if something goes wrong with the code, you can't sue me
 - Require you to include the license if you redistribute my software
 - Technically, you can create your own license, but don't

Adapted from Ben Balter

www.ci.anl.gov

www.ci.uchicago.edu





Licenses



- Generally include "credit me"
- MIT, BSD, Apache (permissive)
 - Explicitly grant nearly unlimited rights
 - Do what you want, including the copyright and permission notices, don't blame/sue the copyright holder if there are problems
- GPL (copyleft, viral)
 - Includes "make source available"
 - derivative works must be GPL (if distributed)
- Dual licensing
 - Usually GPL and something else
 - Users can choose under which terms project is licensed
 - Easier to incorporate within other, already licensed projects

Adapted from Ben Balter





Licensing to Build Communities



- Always license the project
- Coders don't want to give away their code
- Minimize ambiguity, show you speak the language
- Use a familiar license (read: MIT)
 - Include a LICENSE.txt file



Reduce Friction: Increase Contributions



- Make it possible for people to contribute with the least time and effort
 - Reduce acronyms
 - Put a "how to contribute" file in every project
 - Need to show developers how they can contribute, even if they aren't technical/science experts
 - Developers should be able to get the system running on their machine with little effort, e.g. manual dependency management
 - Tell everyone (not just developers) how they can contribute according to whatever they can provide, such as users, those who are interested, testers, educators, evangelists, etc.





Reduce Friction: Increase Contributors



- Make all contributors equal
 - Internal team and external team doesn't work
 - Make all communication electronic and open to all contributors
- Develop/find leaders
 - Put together of a list of people who have been giving the most thoughtful responses – potential leaders
 - When someone invests, invest back
 - Need to keep bringing people in, and finding the leaders
 - Say thank you easy but remarkably effective in encouraging communities
- Make clear, public statements about the community and culture
 - Ask the community to assist with developing these, then ask the community to validate the results
- Help other leaders grow and trust the community





Increase Engagement



- Try failing is learning
- Ask what's working, what's next, what would help, what's missing, would you be willing to get involved to make it happen, who else should be involved
- Communicate Engage more and better with engaging email/posts
- Metrics analyze metrics to guide activities
- Iterate Apply lessons learned, continually

Track engagement activities



- Web views
- E-mails,
- Tickets created
- Tickets closed
- Questions answered
- Documentation created
- Training given
- Code contributions
- Code contributors
- Science discoveries





Gamification



- Ideally used to increase engagement
- Define specific measurable goals
- Bring in new people
 - Give them simple things to do first, and reward them for completing them
 - Welcome them
 - Let them try things out before signing up
- Keep them
 - Personalize interactions
 - Thank them
 - Focus on intrinsic rewards



Conclusions



- Technology for open source is available
- Psychology is even more important
- Small choices can have large effects
 - Think about consequences when naming and describing your project, choosing license, governance, repo/wiki/web, etc.
- Make choices that satisfy intrinsic motivation
- Reduce friction
- Define outcomes, try, measure, change

Questions?



- Now or later
 - d.katz@ieee.org
 - @danielskatz



