

Building a Successful Open Source Community

Alan Turing Institute 4th July 2019



The content of this presentation is available under the terms of the Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license

See <https://creativecommons.org/licenses/by-sa/4.0/> for more information

Modified versions must remove Red Hat trademarks.

About us

“““

Engineering Manager @ Red Hat (7 months)
Software Engineer @ Red Hat (6 months)
Research Associate @NCL (15 years)

Simon Woodman

Engineering Manager | AMQ Streams & Knative Eventing

“““

Senior Software Engineer @RedHat (4 years)
IT jobs @Newcastle University (10 years)
Debian contributor (14 years)
p/t PhD student (CS, 2 years)

Jonathan Dowland

Senior Software Engineer | OpenJDK

Who drives Open Source software?



We know what OS software is but who drives it?


Yes, you can put your software on GH but is that really maximising the value? The maximum bang for your buck?

You tick a few boxes - your boss is happy, the funders are happy.

But who drives the progress in that project? It's the developers and the community

So, what is a community?

community

/kə'mju:nɪti/ 

noun

1. a group of people living in the same place or having a particular characteristic in common.
"Montreal's Italian community"
synonyms: [group](#), [section](#), [body](#), [company](#), [set](#), [circle](#), [clique](#), [coterie](#), [ring](#), [band](#), [faction](#); [More](#)
2. the condition of sharing or having certain attitudes and interests in common.
"the sense of community that organized religion can provide"

5 Reasons to build a community around your project

5 Reasons to **NOT** build a community around your project

1. Time



It takes a lot of time to build the community which obviously takes time away from other things you could be doing

- Publicity
- Helping people
- Documentation, documentation, documentation
- Letting others learn which will take longer than doing it yourself

2. Relinquishing Control



To let the community develop, you have to allow others to have a sense of ownership. We'll talk later about governance models and it's not a case of relinquishing all control but you need to accept you don't have complete control (unless your name is Linus)

- Allow community to influence direction and effort
- Decisions by committee?
- Who can merge PRs?
- What's the makeup of any committees? What happen if people move employers?
- How do you deal with conflict?

3. Lack of Academic Credit



You can't submit a community to the REF. You can't cite a community.

It's somewhat intangible and like the time aspect, it is hard to define how healthy it is.

Large communities change constantly. Boards change, governance changes, ...

As RSEs you can probably get more credit than traditional academics though

4. Lack of Correct Skills



You probably enjoy software development, data science, some kind of scientific discipline if you're in this audience (not wanting to exclude those who don't).

The skills you need to promote, grow and maintain an open source community is different from software development.

Some people can do both, some can't. Growing the community requires soft skills.

Promotion, enthusiasm, inclusion

Not everyone has these skills but in some cases work may be shared around

e-SC - blog not updated for 2 years. I can't regularly tweet about something interesting about it.

5. Chance of Success



Realistically, your chance of building a thriving community is small.

Definitely greater than zero but small. You probably won't be organising events the size of Kubecon next year.

So we should look at what success looks like for your project

However, when we talk about the positive angle later on, a lot of these benefits can be achieved by striving for the community. They are side effects of the journey, not the destination

However...



If you are able to spend the time, do have enough of the skills, don't mind giving up your baby to a group of people who you probably haven't met, know the chance of the baby living is small and don't mind sacrificing your career then listen on...

5 Reasons to build a community around your project



Just by trying to build a community around your project will have significant benefits.

There are thousands, millions of successful open source communities around the world, collaborating and improving the project.

We shouldn't measure success by comparisons with 600lb gorillas such Apache Spark, OpenJDK, ...

And in some cases although those pieces of software have been open sourced, the community is quite closed. The software is successful but not always the community

1. Increased Speed, Reach and Impact



You have limited resources. Build a community and you can accelerate that.

If you build a community people will cite your work, offer co-authorship on papers, funding proposals etc...

Taverna - authors

You show that you are open to collaboration and and interested to work with them

2. Diversified access to skills and experience and kit



Resources not only in people but also in skills and availability of kit / technology

DBZ community contributing DB drivers

Diversified set of use cases

3. Increased lifespan of the project



You start something, it's your baby, you focus your whole attention on it...and then you move on.

Change in fundung, projects, ...

Before I spoke about the chance of success being small. If you can build even a small community around it, the project can live on after you have moved on.

4. Increased career potential



At the extreme end of the spectrum committers on a popular Open Source project are highly sought after and command a very high price. Companies can and do, buy them to influence the direction of the project.

Even if you're not successful, and this may be more industrial focussed, but GH is your CV - it shows engagement and people can look at what you're contributing to

Paolo, Jakub and Tom GH - health?

Even if you try and fail, you will still learn along the way.

Silicon Valley startup culture vs UK

5. Profit

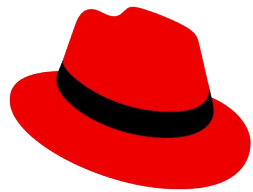


You can make money through Open Source software

- Support
- Training

Red Hat 7 years ago became the first \$1BN open source software company. Shortly it's going to be sold to IBM for \$34BN and it's only IP is the logo. Everything else is open source

Inkspot



Red Hat

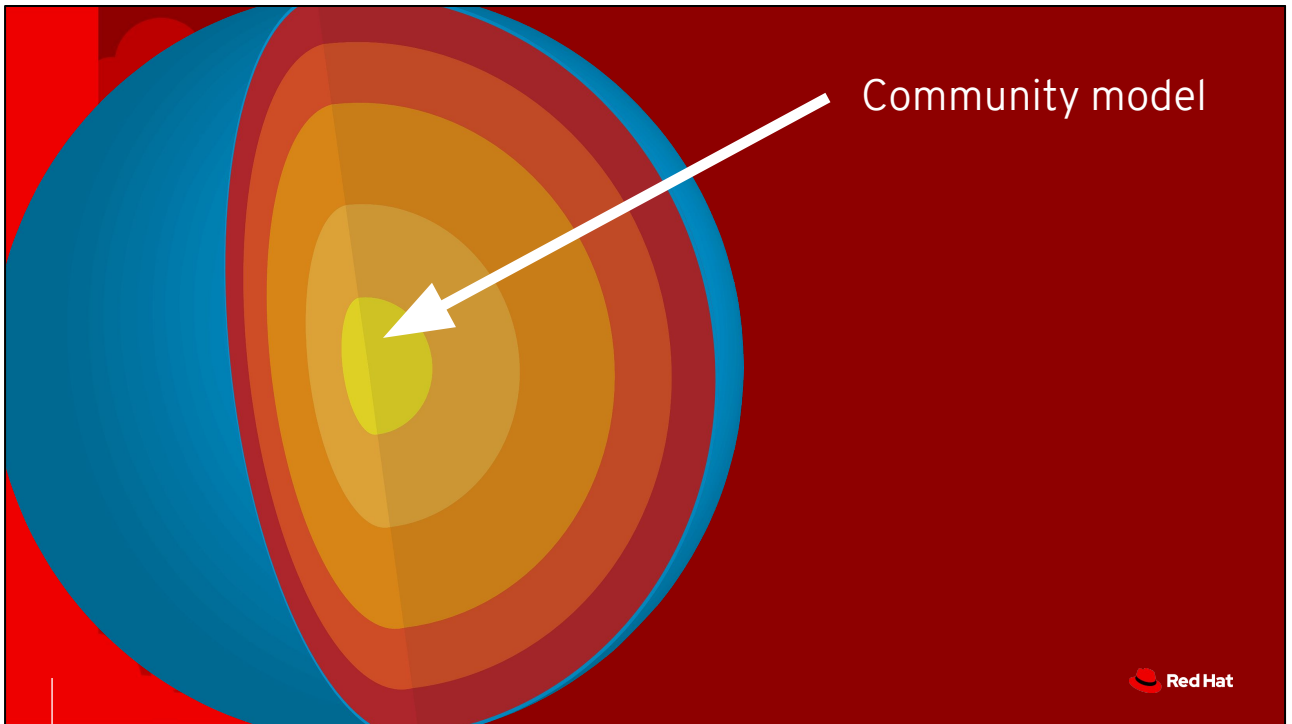
Building a Successful Open Source Community: How

How?

We're going to explore the constituent blocks for bootstrapping a successful open source community and their relative priorities

Firstly, what do you think is the most important thing?

How many of you think the code itself is the most important thing



Argument that the model is more important than the code!

Rules for how the community operate

Including. How the code evolves

Must be self-modifying

“Rules of engagement”

How do you decide which features to add, and when? And how? How do you prioritise which bugs to fix? How do you resolve conflicts when there's more than one way to do things?

Make the implicit, explicit: write down how the project is going to work. Potential contributors are more empowered to contribute if any questions they might have about “how things are done” are easily answered

Hypothetically, given zero code and a community model, you could have a project since you have a process to accept new code.

But the reverse is not necessarily true: you need a decision process for accepting new or modifying code (implicit or otherwise)



“ If you build it,
they will come ”

Stock-photo guy
Some agency, probably

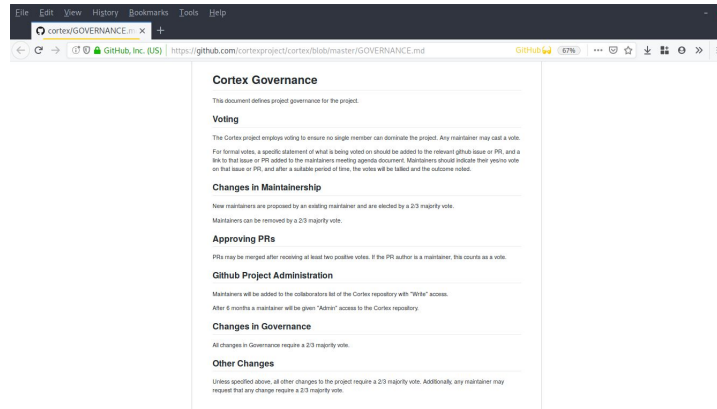
Quote from **Field of Dreams**

Character

“Build it and they will come” is true, but the “it” isn’t your software, it’s your community model.

#0: Community model

Off-the-shelf model



<https://github.com/cortexproject/cortex/blob/master/GOVERNANCE.md>



<https://github.com/cortexproject/cortex/blob/master/GOVERNANCE.md>

An example of a short, self-contained, clear community governance model

An off the shelf model, as opposed to writing one from scratch, will have had some “battle testing”

I was surprised when researching this that I could not find more examples

Three factors to consider



License

- Pick an existing one
- Pick a popular one
- Pick an OSI-approved one

<https://opensource.org/licenses>



Code of Conduct

- Becoming more common
- Recently adopted by Linux

<https://www.contributor-covenant.org/>

1.2.3

Versioning Scheme

- Semantic versioning
- Widely adopted

<https://semver.org/>

Separate talk about licenses coming up

In brief: pick an existing, OSI-approved popular license.

Early in project (hard to change later)

Consider a Code of Conduct

An example of a widely-used one is the “contributor covenant” (itself OSS)

Adopters include the linux kernel

Signals that your project is not a wild-west and potential contributors won't end up in a saloon bar fight

Meaningful well understood semantics for versions

Users Planning upgrades etc

<https://semver.org>

Example Foundations



Apache Software Foundation

- Pick an existing one
- Pick a popular one
- Pick an OSI-approved one

<http://incubator.apache.org/>



Eclipse Foundation

- Becoming more common
- Recently adopted by Linux

<https://www.eclipse.org/org/foundation>



Free Software Foundation

- Semantic versioning
- Widely adopted

<https://www.gnu.org/help/evaluation.html>

Instead of starting your own community, how about trying to join an existing one?

Foundations offer a pre-baked community and often infrastructure and resources
Foundations impose requirements on the software (e.g. license requirements)
GNU requires candidates to not substantially overlap feature-wise with existing GNU programs

Foundations can also offer exposure

Your software may not be mature enough to be worthwhile (yet) submitting to a foundation; but, you could treat that as a goal and work towards it

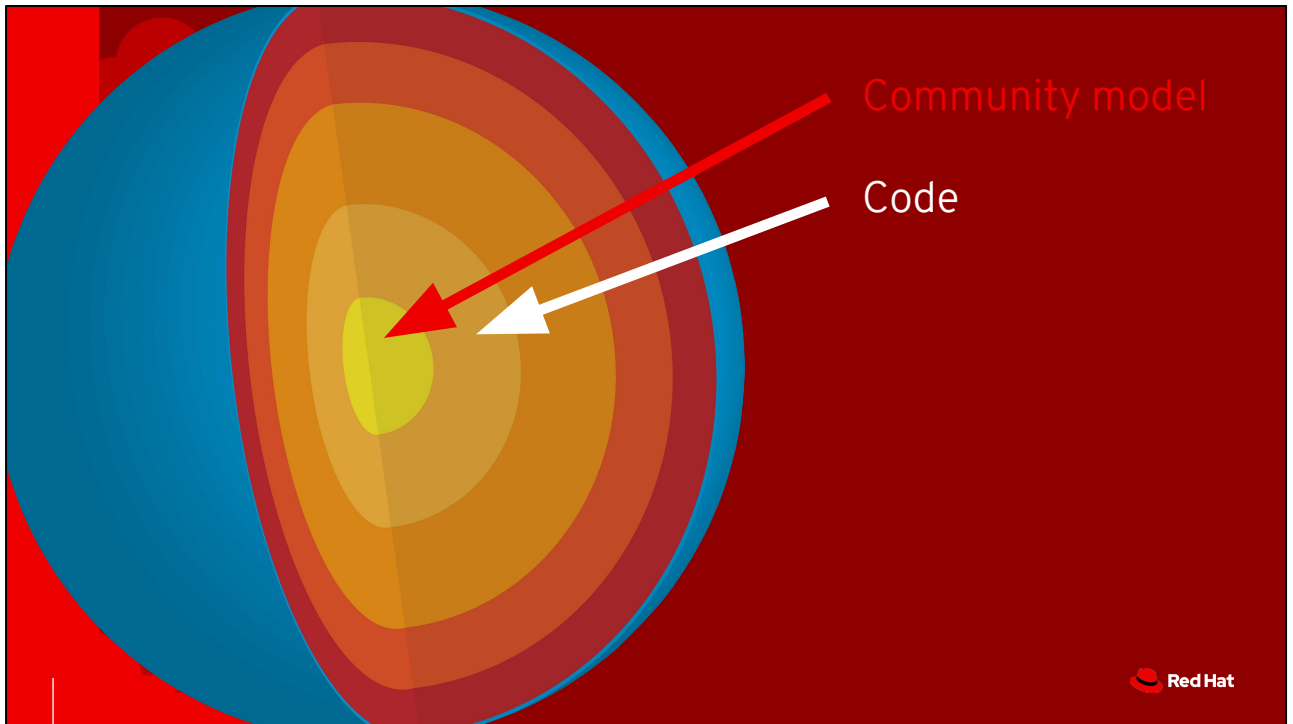
If you are not interested in joining a foundation you may still wish to study them to see what of their governance etc. is worth stealing

Another:

<https://www.freedesktop.org/wiki/>

<https://www.gnu.org/help/evaluation.html>

<https://www.eclipse.org/org/foundation/>

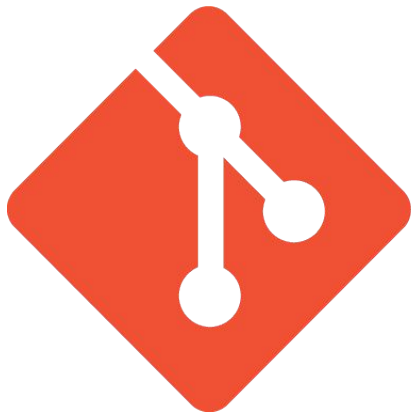


Make the code available!

In the “old days” this would mean creating archive snapshots of the code corresponding to releases and putting them on an FTP server which was part of an FTP mirror network

Archives of release code are still very useful. But people, especially contributors, expect more now. They want access to your version control repository

There are many different version control systems. “Distributed” ones offer many advantages for open source projects over the more traditional centralised ones.



git

<https://git-scm.com/>



Let's cut to the chase: use git.

Git is the runaway success in this space. Use it unless you have a very good reason not to. It's what people will expect to interact with and anything else will be a pain point for people.

(note: OpenJDK uses mercurial)

If you are not yet familiar with VCS, DVCS and GIT respectively, there's a bit of a hill to climb learning the concepts but it's time well invested.

Example Source Code hosting options



GitHub

Very popular
Owned by Microsoft
Private repositories
Issues, Pull Requests
<https://github.com/>



GitLab

“Open Core” - can be self hosted
Private repositories
Issues, Merge Requests

<https://gitlab.com/>



BitBucket

Private repositories
Offers Mercurial hosting

<https://bitbucket.org>



Gitea

Self hosted only
Open Source
Easy “mirroring”

<https://gitea.io>

There are many web sites/services that offer source code hosting via version control repositories for free. The most well known and popular at the moment is GitHub, which is a git-based service

Git hosting
Archives from tags
Issue tracking
Pull requests

Gitlab is “open core” software and you can host your own /adapt your own instances if you want

Gitea is a purely self-hosted open source clone of github. It has a feature that it can mirror external git repositories (and it will automatically periodically fetch new commits)

If you have the means to self-host, even privately, then Gitea or similar may be useful as part of a backup strategy

GitHub (etc.) do occasionally go down!

When to make the initial release?

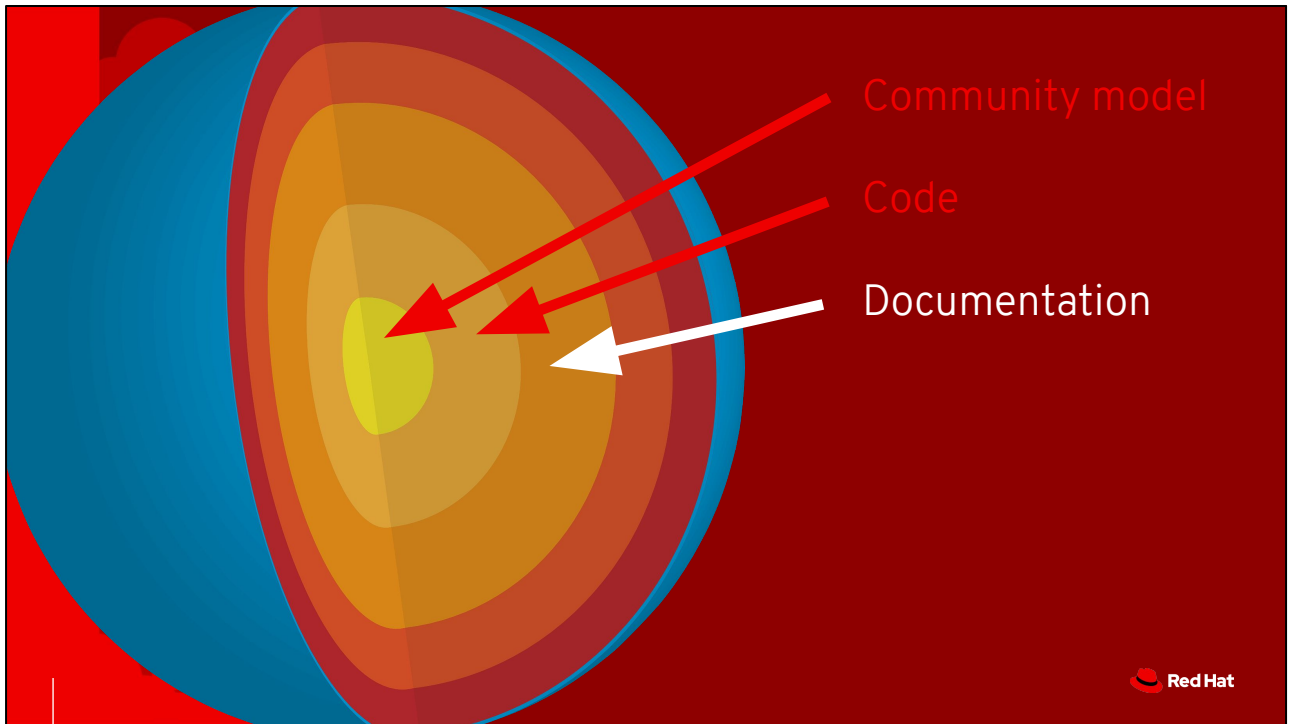


when to initial release? It's important to have something functionally useful (if not "complete" in some sense). People are not going to be drawn to a project that is not practically useful in any way (yet)

GCC 1: initial release entirely by rms:: it compiled code, generated executables (for a single architecture only), bears little resemblance to the later GCC and was inflexible in many ways, but fundamentally it did something useful

Initial linux release did something useful

(with thanks to Andrew Haley)



I've talked about how important the community model is

Documentation is where you actually describe your community model, as well as the code

What is it

How do I use it

FAQ

How do I report bugs with it

I recommend putting the documentation as close to the source code as possible. Ideally, in the same version control repository

There are a number of tools/services that let you generate documentation from a source repository and host it for you

GitHub offers this with "GitHub pages" - you have the source code to a website located in a "docs" folder within the very same repository

Advantages: when someone makes a change to the code, you can insist that they update the docs at the same time, in the same pull request

Larger projects may outgrow this and need to move the documentation out - which introduces synchronisation issues/processes. But cross that bridge as late as possible

Alternatives: readthedocs.io

Wikis

Don't bury the ABOUT page

<https://www.gnu.org/software/hello/> - some example of how to lay out stuff

Yes: README COPYING INSTALL

Maybe: AUTHORS THANKS

I argued No: ChangeLog

Some discussion about ChangeLog from when this was delivered: an "old style" ChangeLog, has effectively been replaced by your VCS history
A ChangeLog is still useful to summarize what has gone into individual releases

Wikis

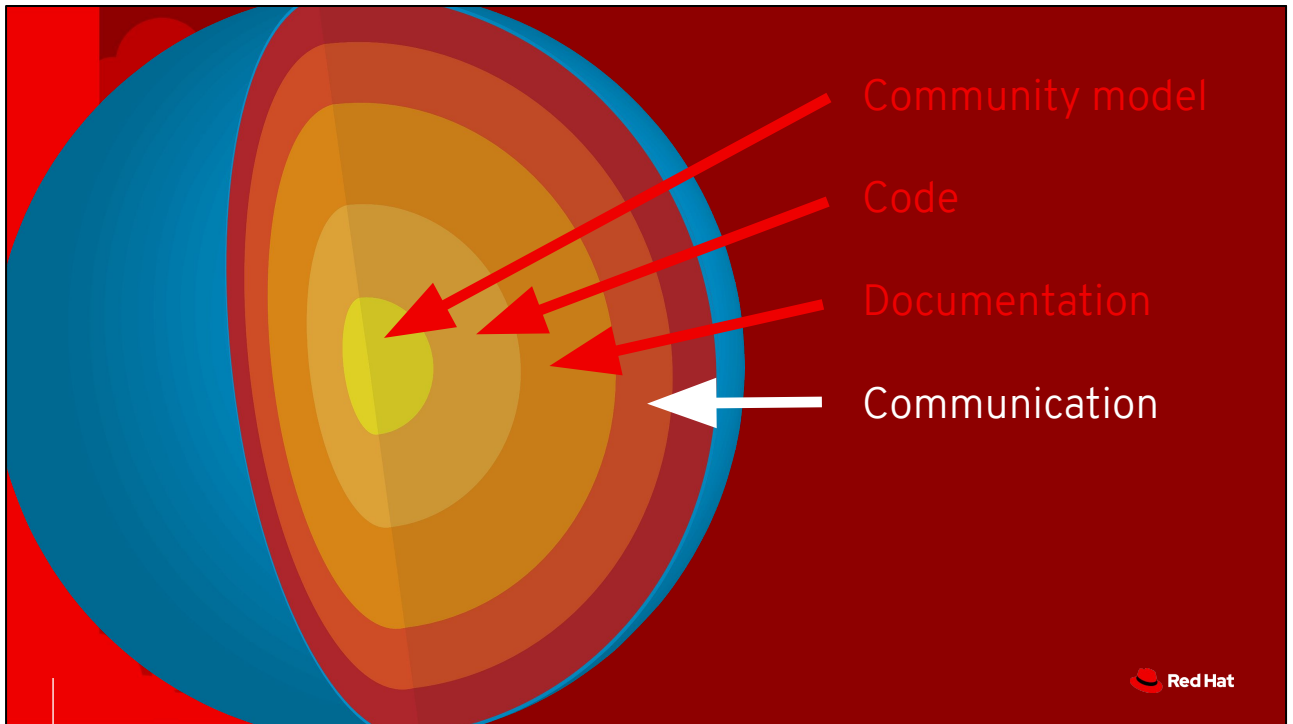


Ultra-low barrier content management system
Anyone can edit; any page can be created etc

E.g. GitHub offer wikis

What makes Wikis interesting is as a communication tool too
(about to talk about communication)

Consider Wikipedia and similar mediawiki projects: they run their governance and do their discussions (the “meta” stuff) within the wiki itself: they do not rely upon an external communication channel for running (most of?) their governance processes.



Mailing lists

Chat

IRC, Slack, etc

Forums

Don't get fixated on a particular solution. A solution is better than none, and there's often no "right" answer

Different types of potential contributors will engage better with different things

Old-timers mailing lists, younger people not so much (written as someone who prefers mailing lists)

Don't worry about having duplication either: two or more solutions seemingly competing with each other; due to the fact different things will play better with different types of contributor this is not necessarily a problem. (It's a high-quality problem: if you are worried about this, it means you've got a community!)

Face-to-face meetings



“ If it’s not on the mailing list, it does not exist ”

Stock-photo guy
Some agency, probably

Risks of in-person meetings: don't be dependent on them or disadvantage people who cannot attend

Try to provide summaries of all discussions that took place, useful information etc., to “catch people up”

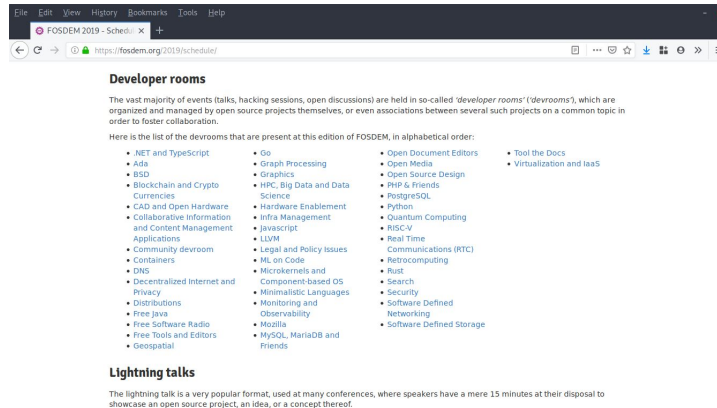
If you organise governance meetings on a real time chat service, make sure its logged and publish the logs, so all decision making is transparent and recorded

Try to avoid disadvantaging contributors who are unable to communicate in real-time

“If it’s not on the mailing list, it doesn’t exist”

“If it’s not on the wiki, it doesn’t exist”

Conferences & FOSDEM



<https://fosdem.org/>



Conferences are a great way to get to meet people face-to-face, even if you are all individually at the conference for different reasons

Exemplar conference for free software:

FOSDEM

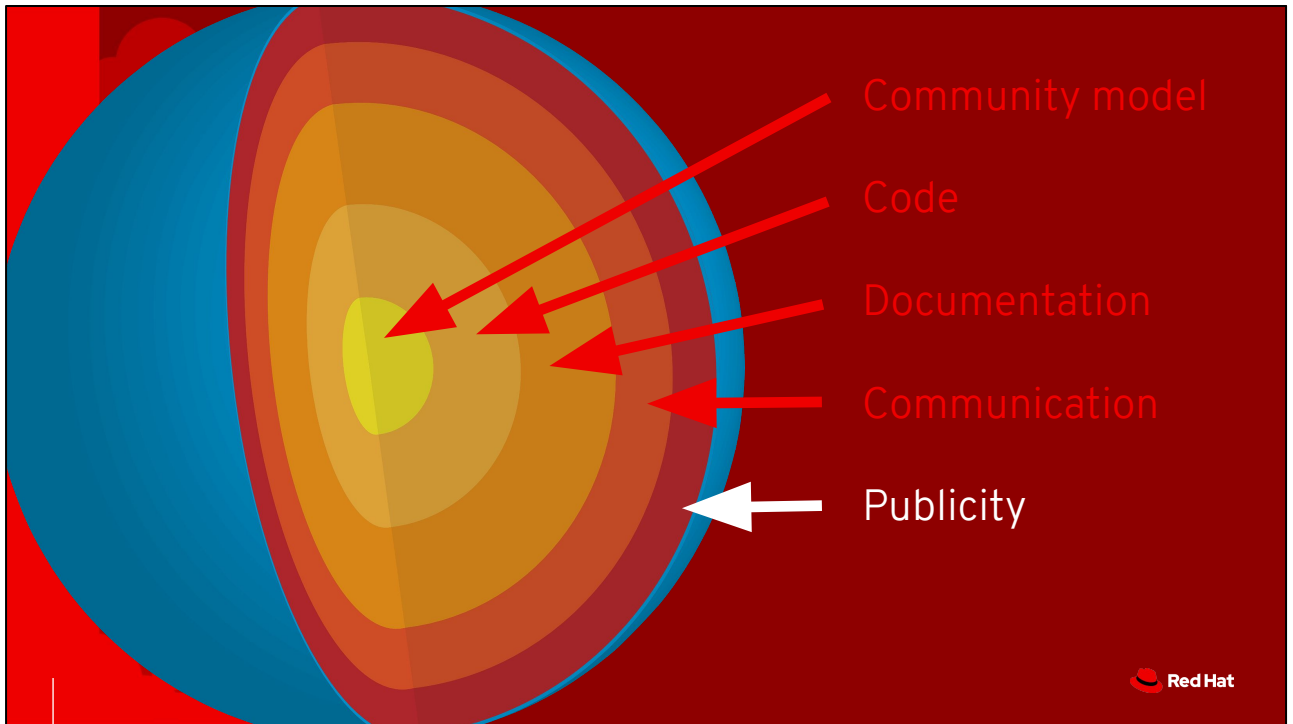
<https://fosdem.org/>

Free to attend

All presentations recorded and distributed for free

Enormous!

(also good for publicity)



The outer shell. It's easy to over-estimate the importance of promoting your open source project at the beginning, but it really is not as important as getting the governance model right, sorting out access to the source code, documentation etc.

The centre of gravity is the community and the code; they will be the strong attractors for contributors

But obviously contributors need to have heard of your project!

Blogs

Twitter, facebook,...

Face to face (probably belong in previous section)

Conferences (probably belong in previous section)

Citations

Blogging & social media



Consider focussing on writing republishable content such as on a blog: this could be picked up and form the basis of articles by tech journalists,
Or be shared on social media directly

The blog could live within your project documentation
Twitter, facebook, Mastodon... instagram?
Changing landscape...

Downstreams



Container registries

<https://quay.io>
<https://operatorhub.io/>
<https://hub.docker.com/>
<https://flathub.org/home>

etc

Distributions

Fedora, Debian, Ubuntu, CentOS (EPEL), Alpine,
Gentoo...
FreeBSD, OpenBSD...

Etc etc



You may end up with “downstream” distributors wanting to distribute your software

That can be great for exposure, more users, etc.

You lose some control: downstreams may adapt your software to meet their goals (subject to their rights under open source license)

It’s often handy to reach out to downstream distributors to establish a personal relationship

Keep an eye on their bug trackers etc.

If you use a distribution or container tool, you could consider preparing packages/container images yourself

If so, bear in mind the norms of those communities may differ from yours (things like: where files go, what your software can do out of the box, etc.)

If you don’t use a distribution or container tool yourself, I would advise you NOT to attempt to produce packages/images for them

It’s easy to do this wrong and cause harm (e.g. packages that interact badly with a distributions native packages, break upgrades etc)

Leave it to someone who does

Citations

► ./parallel

Academic tradition requires you to cite works you base your article on. If you use programs that use GNU Parallel to process data for an article in a scientific publication, please cite:

O. Tange (2018): GNU Parallel 2018, Mar 2018, ISBN 9781387509881,
DOI <https://doi.org/10.5281/zenodo.1146014>

This helps funding further development; AND IT WON'T COST YOU A CENT.
If you pay 10000 EUR you should feel free to use GNU Parallel without citing.

More about funding GNU Parallel and the citation notice:
https://www.gnu.org/software/parallel/parallel_design.html#Citation-notice

To silence this citation notice: run 'parallel --citation' once.



If you are writing papers and relying on software for your work, you could consider citing that software to help raise awareness

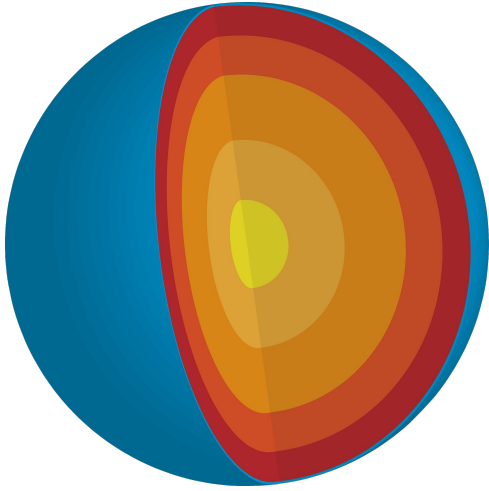
If you are authoring software that you think will be used to support academic work, you could request that people cite your software

Requesting is fine.

Here's an example of what greets you when you first invoke the GNU "parallel" program

My advice is don't do this in the program itself; don't interrupt the use of the program, leave it in the docs

Summary



1. Community model
2. Code
3. Documentation
4. Communication
5. Publicity

Wrap up

Resources

Oss watch - <http://oss-watch.ac.uk/>

...

J. Bacon. The Art of Community: Building the New Age of Participation . O'Reilly Media, 2012. <https://www.ionobacon.com/books/artofcommunity/>

<https://community.redhat.com/>

<https://www.gnu.org/software/hello/> – GNU Hello World

<https://github.com/cortexproject/cortex/blob/master/GOVERNANCE.md> (off-the-shelf community model starter)

<http://incubator.apache.org/guides/proposal.html>

<https://www.gnu.org/help/evaluation.html>

<https://www.eclipse.org/org/foundation/>



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

 [linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)

 [youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)

 [facebook.com/redhatinc](https://www.facebook.com/redhatinc)

 twitter.com/RedHat