Open Source Introduction

Contribution, Management, People ...

PB173, Milan Broz <xbroz@fi.muni.cz>

Open Source / Free Software

It's free as in freedom – think free speech, not free beer.

Open source is "**Culture by choice**"

- GNU definition: https://www.gnu.org/philosophy/free-sw.html
- OSI definition: https://opensource.org/osd

Open? Free? It means ...

- Zero-cost software?
- Right to use, modify or even fork source code?

  Without releasing changed source code?

  Even in commercial or proprietary projects?

It depends => Choosing appropriate License

History: E.S.Raymond – The Cathedral and the Bazaar

Recent: K.Fogel – Producing Open Source Software, https://producingoss.com/

Licenses

- **License examples** (code / documentation)

  GNU GPL, GNU FDL, CC, MIT license, BSD, Apache, ...
  https://www.gnu.org/licenses/license-list.html

- **Early decision**

  Change later (often impossible) – all contributors must agree

- **CLA – Contributor License Agreement**

  Required in some projects

  Example: OpenSSL, https://www.openssl.org/policies/cla.html

- **Transfer of Copyright**

  Example: FSF – Free Software Foundation projects
  https://www.gnu.org/prep/maintain/html_node/Copyright-Papers.html

- **(US) Patents – special license clauses**

Proprietary vs Open Source

Open source projects (usually):

- **Release (code) early, release often**
  ... in reality, it depends on project authors attitude
- **If there's no reason for it to be private, it should be public**
  ... in reality, sometimes decision behind closed doors
- **Cannot manage developers directly**
  Compare: employee in a company versus independent contributor
- **Forks**
  Anyone can fork code and start own derived project
  The problem is the loss of users and developers, not the fork itself
- **Benevolent dictator model** (final decision = one person)
- **Consensus based decision (voting, discussion)**
- **Community**
  ... in reality, who forms the community?

Proprietary vs Open Source

Close source and proprietary software

- **Common for "mainstream" companies and corporations**
  - Open source is taken seriously internally
  - But often just as a threat (to revenues)
- **Rigorous project planning and management**
  - Release plan, milestones… and failures ☺
  - "Firm" deadlines (promises to customer => money)
- **Market share, competition**
- **Intellectual property protection**
- **Decision behind closed doors**

Could [project] work for us for free?
How to monetize "free" users?
...
Sponsoring projects, conferences.
Contributing to project directly (code)
or indirectly (allow access to specific hw,
build farms for testing).

Copyright, Trademarks, Patents

- **NDA – Non-Disclosure Agreement**
  - *Protect confidential, proprietary or trade secret information*
- **Improper use of copyrighted code, trademarks**
  - *Can be fixed by removal, rewrite or rename of the project*
- **Patent encumbered ideas (US patents)**
  - *Cannot be fixed*
    Use defensive thinking to avoid this problem in the first place
  - *Expensive lawsuit is usually not the option*
  - *Neither the license for a patent use*
  - *Note Red Hat patent promise for Open Source Software*
    https://www.redhat.com/en/about/patent-promise

Project management, people and roles

- **Upstream** -> downstream: distributions, releases (own maintainers)
- **Small project** – one person + few contributing members
- **Large projects**, more roles (*... in theory*), usually combined
  - Project lead (or committee)
  - Developers, commiters
  - Code reviewers
  - QA & Test developers
  - Bug triage
  - Mailing list, wiki, IRC, social network administrators
  - Release handling
  - Documentation and translation

Infrastructure & Tools

Have no fear of perfection, you'll never reach it. – Salvador Dali

- **SCM – Source Code Management**
  - Use **git** today, even for local and small projects
  - History, branches, merge of contributions
  - Tags (generated releases), bisection (bug hunting), …
- **Bug / Issue tracker** (JIRA, Bugzilla, GitHub issues)
  - Allow easy bug reports (no complicated registration)
  - Delay between upstream release and bug reports
  - Active use (users, developers)
- **Mailing list**
  - Announces, discussions, bug reports
- **All-in-one solution**
  - GitHub and GitLab are popular today

Infrastructure & Tools

In anything at all, perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away. – Antoine de Saint-Exupery

- **IRC, Wiki, Social networks**
  - Nice to have but require active maintenance
- **CI: Continuous integration** (BuildBot, Jenkins)
  - Also CI/CD – Continuous Integration & Delivery (= Deployment)
  - Build farms
  - Regression testing, test frameworks
  - Performance testing, stable API
  - Without good testsuite it is waste of effort (actively maintain tests)
- **External code quality tools**
  - Static analysis (Coverity and similar)
- **Review tools**

Documentation

- **Release documentation**
- **FAQ – Frequently Asked Questions**
  - Useful in discussion – direct link to an answer
- **API documentation**
  - Can be generated (Doxygen or similar tool)
  - API use examples
  - API stability
- **Manual pages**, online manuals
- **Code style, code formatting guides**

Communication (& Politics)

Have You Tried Turning It Off And On Again? – The IT Crowd

- **Your project must appear alive, communication is a must**
- **Building trust takes long time**
- **You are what you write**
  - Mailing list archives, chat logs, commit messages are public
  - Many people will search information about you
- **No need to respond to everything**
  - Successful project has users (= Community) handling a lot of questions
- **Avoid ad hominem arguments**
  - It is ~~almost~~ always ad hominem fallacy
- **Use emotions with care**
  - Make apologies if needed (nobody is perfect)

Communication

(& Psychology)

- **Parkinson's law of triviality**
  - Unproductive discussions
  - **Bikeshedding**, http://bikeshed.com

Parkinson shows how you can go in to the board of directors and get approval for building a multi-million or even billion dollar atomic power plant, but if you want to build a bike shed you will be tangled up in endless discussions.

- **Trolling**
  - Upsetting people by using extraneous or off-topic arguments
- **Be honest**
  - Even the most boring question can uncover very interesting problem
  - If abusing lists, link to FAQ helps (... students & easy lab solutions ;-)
  - Different point of view prevents tunnel vision
- **Multicultural environment**
  - Sarcasm, irony and humor can be understood differently
  - *But it is your project, your work and your fun :-)*

Communication

(& Psychology)

- **Happy users are usually quiet**
  - But bug reports is excellent metric for project success
- **Difficult people**
  - They can be excellent developers with poor social skills (or even personality disorders)
  - You will lose many excellent ideas if you just ignore them
- *In **extreme cases** remember **Dunning-Kruger effect***
  http://rationalwiki.org/wiki/Dunning-Kruger_effect

Bad Communication ...

Following few examples are kind of thought-provoking.

They are lift out of context intentionally.

Bad Communication ...

excellent contribution to code vs ad hominem arguments

> Have you read it? Once again, it is about IPv6. [...]
Everything, but really everything, you say is complete garbage.
People like you are the reason I try my hardest to avoid having anything to do with Fedora development.
Go, dig a hole and sit in it. It's a more worthwhile use of your time.

–
Ulrich Drepper, 2007
*[**lead contributor and maintainer of glibc** (GNU C library)]*

http://www.redhat.com/archives/rhl-devel-list/2007-October/msg01073.html

Communication ...

Linux kernel list (in the past)

There are a number of very good Linux kernel developers,
but they tend to get outshouted by a large crowd of arrogant fools.
Trying to communicate user requirements to these people is a waste of time.
They are much too 'intelligent' to listen to lesser mortals.

– Jack O'Quin, Linux audio developer

http://lwn.net/Articles/131776/

Note

- Most of the communication is very friendly.
- Volume of the kernel list is extreme high (hundreds of posts per day).

Communication ...

”Old” Linus' style (sometimes)

Dmitry Kakurin wrote:
> When I first looked at Git source code two things struck me as odd:
> 1. Pure C as opposed to C++. No idea why.
> Please don't talk about portability, it's BS.
**YOU* are full of bullshit.*
*C++ is a horrible language. It's made more horrible by the fact that a lot of substandard programmers use it, to the point where it's much much easier to generate total and utter crap with it. Quite frankly, even if the choice of C were to do *nothing* but keep the C++ programmers out, that in itself would be a huge reason to use C.*
...
Linus Torvalds, 2007
http://harmful.cat-v.org/software/c++/linus

- Surprisingly, strong words help to find a quick way to fix problems.
  But there are better ways!
- Also a nice example starting a flame unrelated to the git project.

Communication ...

>> "Mauro, SHUT THE FUCK UP!"
>
> This one crosses the line. There's no non-offensive way to tell a geek
> "you are wrong", but this isn't even trying. Bad Linus!

You know what? Not my proudest moment. I was really upset.
...
Neil Brown here somewhere earlier said
"So my personal perspective on what it means to be responsible is:
Don't flame: include the facts, exclude the emotion."
and I can't overstate how much I disagree. You do need the factual
part too, but "exclude the emotion" is not good either.
...
Linus Torvalds, 2013
https://lwn.net/Articles/559178/, also read https://lwn.net/Articles/559061/

Since 2018, Linux kernel code of conduct to improve contributions culture:
https://www.kernel.org/doc/html/latest/process/code-of-conduct.html

OSS project examples

(projects of various scopes from small to large)

- **Util-linux – https://github.com/karelzak/util-linux**
  - Large set of utilities for Linux
  - Many contributors, one maintainer
- **OpenSSL – https://www.openssl.org/**
  - Widely used cryptographic library
  - Many contributors, small group of maintainers, CLA required
- **Ceph – https://ceph.io/**
  - Distributed storage platform based on object store
  - Chief architect, maintainers, The Ceph foundation (industry members)
- **Linux kernel – https://www.kernel.org/**
  - One of the biggest OSS projects
  - One maintainer, several sub-tree maintainers, many contributors

Q/A