Producing Open Source Software

- How to Run a Successful Free Software Project -

2016.

정성인, ETRI

Book overview

- By Karl Fogel
- 2005-2016
- Based on the experience of Subversion project

Preface

- Most free software projects fail
  - Failure is not an event
  - There is not even a clear definition of when a project is expired
- Free and open source
  - Zero-cost software
  - The freedom to share and modify for any purpose

Getting started

- How to introduce a new free software project to the world
  - But first, look around
    - Might find something so close
    - Exceptionally, educational experience, pre-exiting code won’t help
  - Found that nothing
    - Private vision into public vision
    - Every good work of software starts by scratching a developer’s personal itch

Getting started

- Starting from what you have
  - Choose a good name
  - Have a clear mission statement
  - State that the project is free
    - On the front page, unambiguously clear
  - Feature and requirement list
  - Development status
    - Launch-pad.net
  - Download
  - Version control and bug tracker access
    - A sign that please join and help
  - Communication channels
  - Developer guidelines
  - Documentation
  - Demo, screenshots, videos, and example output
  - Hosting
    - Canned hosting

Getting started

- Choosing a license and applying it
- Setting the tone
  - Avoid private discussions
    - Discussion will help train and educate new developers
    - Massive peer review
  - Nip rudeness in the bud
  - Code of conduct
  - Practice conspicuous code review
  - Be open from day one
    - *Technical debt*
  - Waiting just creates an exposure event
- Announcing
  - news.ycombinator.com, openhub.net, openhatch.org,…..
  - Subversion and Mozilla project were launched without running code

Technical infrastructure

- *Everyone feel like they're all working together in the same room*
- what a project needs
  - Web site
    - Canned hosting
  - Mailing list and message forums
    - Bread and butter of project communication
  - Version control
    - Version everything
  - Bug tracker
    - Not only bug reports, but new feature requests, one-time tasks, unsolicited patches
    - Interaction with email
    - Pre-filtering the bug tracker
  - IRC
    - IRC bots
  - Wikis
  - Q&A forum
  - Translation infrastructure
  - Social networking services

Social and political infrastructure

- *How does it work? What keeps a project running? Who makes the decisions?*
- “successful”
  - Not just technical quality, operational health and survivability
- a formal governance structure or less formal structure, but more self-restraint
- Forkability
  - the ability of anyone to take a copy of the source code and use it to start a competing project
  - Implies consensus
- Benevolent Dictators(BD) model
  - Do not actually make all the decision, or even most of the decisions
  - It is unlikely that one person could have enough expertise to make consistently good decision across all areas of the project
  - Quality developers won’t stay around unless they have some influence on the project’s directions
  - Let things work themselves out through discussion and experimentation whenever possible
  - One when it is clear that no consensus can be reached, most of the group wants someone to guide the decision
  - Who?
    - Not need to have the sharpest technical skills of anyone in the project

Social and political infrastructure

- Consensus-based democracy
  - When consensus cannot be reached, vote
  - When to vote
    - it ends discussion, and thereby ends creative thinking about the problem
    - As long as discussion continues, there is the possibility that someone will come up with a new solution
    - After a compromise, everyone is a little bit unhappy. Whereas after a vote, some people are unhappy while others are happy
  - Veto
    - Any veto should be accompanied by a thorough explanation
    - -1 (from Apache software foundation)

- Writing it all down
  - Need to record it somewhere
  - Libreoffice development guide, Subversion community guide, Apache software Foundation governance document

Social and political infrastructure

- Joining or creating a non-profit organization
  - Successful open source projects often get to a point where they feel the need for some of formal existence as a legal entity – to be able to accept donations, to purchase and maintain infrastructure for the project's benefit, to organize conferences and developer meetups, to enforce trademarks, etc
  - One of the reasons to join one of the existing organizations is that
    - It provide a legal home for open source projects
    - It gives economies of scale and broad experience
    - they already have experience with this distinction
    - Know how to fairly read the will of the project even when there is controversy or strong disagreement
    - Also serve as a neutral place for resolving disagreement
    - Project psychotherapist
    - Some well-known and reputable umbrella
      - Software freedom conservancy, Apache software foundation, Eclipse foundation, OuterCurve foundation, Software in the public interest, Linux foundation
  - If your project joins or creates a non-profit organization, make clear that the separation between the legal infrastructure and the day-to-day running of the project

Organizations, money, and business

- How to use money constructively in a free software environment
- Is not primarily about to how to find funding source for your open source project
- For reference, https://en.wikipedia.org/wiki/Business_models_for_open-source_software

- The economics of open source
  - Most free software is written by paid developers, not by volunteers
  - A company often needs a particular piece of software to be maintained and developed, and yet does not need a monopoly on that software
    - Electric grids, roads, sewer systems, and other goods that everyone needs, but no one needs to own
  - Informal(volunteers, subsidy) -> formal, and corporations have become conscious of the benefits of open source software
  - Financial backing is generally welcomed by open source development communities
    - If you hire good programmers, and they stick around long enough to get experience with the software and credibility in the community, then they can influence the project by the same means as any other member

Organizations, money, and business

- Typed of corporate involvement
  - Sharing the burden
  - Augmenting services
  - Creating an ecosystem
  - Supporting hardware sales
  - Undermining a competitor
  - Marketing
  - Proprietary relicensing
  - Donations

- Governments and open source
  - Governments differ from individuals and from private-sector organizations in some fundamental ways
    - Governments often aren't trying to retain technical expertise in-house
    - Governments have labyrinthine and inflexible procurement and employment policies
    - Government agencies tend to be unusually risk-averse
    - Government officials hunger for well-timed and well-controlled publicity events
    - 18f.gsa.gov, Mil-OSS.org, dwheeler.com, ben.balter.com/2015/11/23/why-open-source

Organizations, money, and business

- Hire for the long term
  - A long-time developer also knows all the old arguments that have been hashed and rehashed on the discussion lists
- Appear as many, not as one
- Be open about your motivations
  - Be as open about your organization's goals as you can without compromising business secrets
- Money can’t buy you love
  - Example, CollabNet hired a skilled developer for Subversion and gave a right to commit directly, but non-salaried developers feel disenfranchised
- Contracting
  - Contracted work needs to be handled carefully in free software projects
  - It's very hard to produce an acceptable patch for a non-trivial enhancement or new feature while truly being a complete stranger
- Funding non-programming activities
  - QA, legal advice and protection, documentation and usability, providing hosting and bandwidth, providing build farms and development servers, and sponsoring conferences/hackathons/other developer meetings

Organizations, money, and business

- Marketing
  - How to avoid common pitfalls in marketing open source products
  - Open source and freedom from vendor lock-in
  - Remember that you are being watched
  - Don’t bash competing open source products
  - Don’t bash competing vendors’ developers
  - Commercial vs. proprietary
- Open source and the organization
  - Dispel myths within your organization
    - *If it’s open, that means anyone can change our code, open source software is insecure, because anyone can change the code, open source comes with no support, open source is cheaper(all of those other costs are, on average, the same for open source software. And it is often cheaper in the long run because it frees your organization from proprietary vender lock-in), if we open source this project, we will have to spend a lot of time interacting with outside developers, if we open source this project, then we’ll have to release all our other stuff as open source too, developers will devote attention to this code just because we released it, other companies will pic up this software and start using it right now*
  - Foster Pools of Expertise in Multiple Places
  - Don’t let publicity events drive project schedule
  - The Key Role of Middle Management
  - Innersourcing : but not the same as open source

Organizations, money, and business

- Hiring open source developers
  - The kind of activity to look for is not only their direct technical activity, but also their relations with the other developers in the project
  - Visit a threaded view of the project's discussion forums
- Evaluating open source projects
  - Is certainly an ay, not a science
    - Look at bug tracker activity first
    - Measure commit diversity, not commit rate
    - Evaluate organizational diversity
    - Discussion forums
    - News, announcement, and release
- Crowdfunding and bounties
  - Bounties are one-time rewards for completing specific tasks, such as fixing a bug or implementing a new feature
    - https://www.bountysource.com/
  - Both crowdfunding and bounties have funded some open source work, they have not been a major economic force compared to salaried or contracted development

Communications

- A great programmer with lousy communications skills vs. a lousy programmer with good communications skills
- How to conduct your own communications, and how to make maintenance of communications mechanism a priority for everyone in the project

- You are what you write
  - Consider : the only thing anyone knows about you on the Internet comes from what you write, or what others write about you
  - Story of FSF GNU Emacs
  - Structure and formatting
  - Content
    - Make things easy for your readers
    - Don’t engage in hyperbole
    - Edit twice
  - Tone
  - Recognizing rudeness
  - Face
    - Email name is your online face

Communications

- Avoiding common pitfalls
  - don’t post without a purpose
  - Productive vs. unproductive threads
    - On a busy mailing list, one is to figure out what you need to pay attention to and what you can ignore. The other is to behave in a way that avoids causing noise.
  - The smaller the topic the longer the debate
    - Bikeshed effect
  - Avoid holy wars
    - In a holy war, everyone knows there is one right answer: they just don’t agree on what it is
    - Once a holy war has started, it generally cannot be resolved to everyone’s satisfaction
    - So how should you handle holy wars?
      - First try to set things so they don’t happen
      - Express sympathy and understanding
      - Reciprocal gesture
      - When can’t be avoided, give up(back out)
  - The noisy minority effect

Communications

- Difficult people
  - Not mean rude
  - Handling difficult people
- Choose the right forum
  - One of the trickiest things about managing an open source project is getting people to be thoughtful about which forum they choose for different kinds of communications
  - IRC is terrific for quick questions, but not a good forum for making decisions that affect the whole project
  - Mailing list is a great place for making formal project-wide decisions
  - Bug tracker, so well integrated with email. Discussion on a bug ticket
  - You have to create a culture where it’s normal fir everyone to do it
- Publicity
  - Announcing releases and other major events
  - Announcing security vulnerabilities

Packaging, releasing, and daily development

- Highway repair

- Release numbering
  - Some projects just need release identifiers, not release numbers
    - Release numbers are most important for people who don't follow the project on a daily basis
  - Release number components
    - Major number, minor number, micro number, {patch number}
      - Scanley 2.3.0(Alpha 1)
  - The simple strategy
    - Changes to the micro number : both for forward and backward compatible. Bug fixes only or very small enhancements to existing features. No new features
    - Changes to the minor number : backward compatible. It's normal to introduce new features in a minor release, but usually not too many new features at once
    - Changes to the major number : forward and backward incompatible. Is expected to have new features
  - The even/odd strategy
    - Even means stable, odd means unstable
    - This applies only to the minor number, not major or micro numbers
    - Best for projects that have very long release cycles

Packaging, releasing, and daily development

- Release branches
  - A free software is in a state of continuous release
  - Full-tree snapshots for releases is impossible
  - Release branch : the release can be isolated from mainline development
  - Release branches are pretty much required in open projects, however, I have seen projects do releases without them, but it has always resulted in some developers sitting idle
- Stabilizing a release
  - Time-based release vs. feature-based release
  - The process of stabilizing a release is mostly about creating mechanisms for saying “no”
    - Low-risk or non-core change, fixes for severe bugs, documentation updates
  - Release owner(not be the same person as the project leader)
    - Release owner needs the technical competence and people skills
    - "I don't think there's anything wrong with this change, but we haven't had enough time to test it yet, so it shouldn't go into this release.“
    - Release manager
      - keep track of how many changes are currently under consideration, how many have been approved, how many seem likely to be approved. etc
  - Voting on changes
    - At the opposite extreme from release owner, developers can simply vote

Packaging, releasing, and daily development

- Packaging
  - The canonical form for distribution of free software is as source code
  - Format
  - Name and layout
    - the software name + the release number + format suffixes. ex) scanley-2.5.0.tar.gz
    - when unpacked, a single new directory tree named scanley-2.5.0 must be created
    - in the top level of new directory, there should be a plain text : README, INSTALL, LICENSE(COPYING), CHANGES(NEWS)
  - Compilation and installation
    - For software requiring compilation or installation from source, there are usually standard procedures that experienced users expects to be able to follow
  - Binary packages

- Testing and releasing
  - Release checklist criteria
    - Regression testing, digital signature, …
  - Candidate releases and announcing releases
    - Wide testing before official release

Packaging, releasing, and daily development

- Maintaining multiple release lines
  - most mature projects maintain multiple release lines in parallel
  - ”support” means accepting bug reports against that line

- Planning releases
  - One area where open source projects have historically differed from proprietary projects is in release planning
  - Conflicts between the priorities of those developers who are being paid and those who are volunteering their time. These conflicts often happen around the issue of when and how to schedule releases
  - There is no general solution to this problem except discussion and compromise
    - By nailing down features sets early, you reduce the complexity of the discussion
  - The other way the project can lower tensions around release planning is to make releases fairly often(eg, every three and six months)

Managing paricipants

- Community and motivation
  - Why do people work on free software projects?
    - My job(my manager asked)
    - Want to produce good software
    - Want to be personally involved I fixing bugs that matter to them
    - Challenge and educational value of working on hard problems
    - A build-in desire to work with other humans
  - Delegation
    - Is not merely a way to spread the workload around. It is also a political and social tool for drawing people into a closer commitment to the project
  - Praise and criticism
  - Prevent territoriality
    - Cookie licking
    - In order to combat incipient territorialism, many projects have taken the step of banning the inclusion of author names or designated maintainer names in source files
      - *At the Apache Software foundation, we discourage the user of author tags in source code. There are various reasons for this, apart from the legal ramifications. Collaborative development…*
  - The automation ratio

Managing paricipants

- Community and motivation
  - The automation ratio
  - Treat every user as a potential participant
  - Meeting in person
    - Social connections
    - Don’t sponsor a hackathon with just the limited goal of getting a specific list of bugs fixed or features implemented
    - Deeper long-term collaboration is the goal: the event is just a means of getting there
- Share management tasks as well as technical tasks
  - Manager does not mean owner
    - Responsibility without monopoly
    - Each domain manager’s job is to notice when other people are working in that domain, and train then to do the things the way the manager does, so that the multiple efforts reinforce rather than conflict
    - Domain managers should also document the processes by which they do their work, so that when one leaves, someone else can pick up the slack right away
    - Patch manager, translation manager, documentation manager. Issue manager,…
- Transitions
- Committers

Managing paricipants

- Credit
  - Reputation
  - Monetary value
  - To be appreciated
  - One of the most important features of collaborative development software is that it keeps accurate records of who did what, when
  - Guidelines
    - Thanking someone for their bugfix in passing during an IRC conversation is fine. Don’t post a new email solely to thank someone
    - Don’t clutter the project’s web pages with expressions of gratitude
    - Never put thanks into comments in the code

Legal matters: Licenses, Copyrights, Trademarks and Patents

- The project's license is compatible with its goals
- Terminology
  - Free software
    - A social movement
  - Open source software
    - A development methodology rather than as a political movement
  - FOSS, F/OSS, FLOSS
  - DFSG-compliant
    - https://wiki.debian.org/DFSGLicenses
  - OSI-approved
    - https://opensource.org/licenses
  - Proprietary, closed-source
    - Commercial is used as a synonym for proprietary, but the two are not the same
  - Public domain
    - Having no copyright holder
  - Copyleft
    - A license that not only grants the freedoms under discussion here but furthermore requires that those freedoms apply to any derivative work
  - Non-copyleft or permissive
    - A license that grants the freedoms under discussion here but that does not have a clause requiring that they apply to derivative works as well
    - BSD, MIT licenses

Legal matters: Licenses, Copyrights, Trademarks and Patents

- The GPL and license compatibility
  - FSF maintains a list showing which licenses are compatible with the GPL
  - http://www.gnu.org/licenses/licenses-list.html

License and specific version	Compatible to the GPL
Academic Free License	No
Apache License version 1	No
Apache License version 1.1	No
Apache License version 2	Yes[2]
Apple Public Source License version 1.x	No
Apple Public Source License version 2.0	No
Artistic License 1.0	No
Clarified Artistic License (draft 2.0)	Yes
Artistic License 2.0	Yes
Berkeley Database License	Yes
original BSD license	No
modified BSD license	Yes
Boost Software License	Yes
Common Development and Distribution License	No
Common Public License	No
Creative Commons licenses (Tags: by & sa)	No
Creative Commons licenses (Tags: nc & nd)	No
Cryptix General License	Yes
Do What The Fuck You Want To Public License (WTFPL)	Yes
Eclipse Public License	No
Educational Community License	?
Eiffel Forum License version 2	Yes
GNU General Public License	Yes[3]
GNU Lesser General Public License	Yes

License and specific version	Compatible to the GPL
Hacktivismo Enhanced-Source Software License Agreement	No
IBM Public License	No
Intel Open Source License	Yes
ISC license	Yes
LaTeX Project Public License	No
MIT license	Yes
Mozilla Public License	No
Netscape Public License	No
Open Software License	No
OpenSSL license	No
PHP License	No
POV-Ray-License	No
Python Software Foundation License 2.0.1; 2.1.1 and newer	Yes
Q Public License	No
Sun Industry Standards Source License	No
Sun Public License	No
Sybase Open Watcom Public License	?
W3C Software Notice and License	Yes
XFree86 1.1 License	Yes[2]
zlib/libpng license	Yes
Zope Public License version 1.0	No
Zope Public License version 2.0	Yes

Legal matters: Licenses, Copyrights, Trademarks and Patents

- Choosing a license
  - Use an existing license instead of making up a new one
    - Are familiar to many people already(people won't feel they have to read the legalese in order to use your code)
  - GNU GPL v3
  - GNU AGPL v3
  - MPL
  - LGPL v3
  - EPL 1.0
  - MIT
  - Apache license 2.0
  - BSD-2
  - Descending order from strong copyleft at the top to completely non-copyleft at the bottom

- Contributor agreements
  - Doing nothing
  - Contributor license agreement(CLA)
  - Copyright assignment(CA)

Legal matters: Licenses, Copyrights, Trademarks and Patents

- Proprietary relicensing
  - Some companies offer open source code with a proprietary relicensing scheme
    - The first kind is about selling exceptions, the second one is freemium or open core model
  - vs. dual license
  - Problems with proprietary relicensing
    - It discourages the normal dynamics of open source projects
    - Shakedown model

- Trademarks
  - Trademark law as applied to open source projects does nc g from trademark law as applied elsewhere
  - Mozilla Firefox, the Debian project, and Iceweasel
  - The GNOME Log and the Fish Pedicure shop

- Patents
  - Software patents should be abolished entirely
  - Only to used as a defense

Thank you