Universidad Rey Juan Carlos

Master in Libre Software

Case Studies

How goals are handled by selected FℓOSS projects

Author:
Daniel H.Gámez V.

Lecturer:
Gregorio Robles

April 24, 2014

Abstract

The following report is focused on the goals of the selected projects, including information from as many of the projects that have been presented in this subject as possible. A group of talks presented by people involved in FLOSS will show the projects vertically, providing information regarding the history, the goals, the members, the licenses, the related industry, the technologies, the governance, among others.

Contents

1 Video Talks

1.1 How the FreeBSD Project Works

This talk was given by Robert Watson from the University of Cambridge Computer Laboratory, and took place on 2007 at the Google TechTalks meeting. As a current kernel developer Watson has been actively involved in the FreeBSD project since 1999. The talk is about the process of writing Open Source in a large FLOSS project.

1.1.1 Brief description of FreeBSD

FreeBSD is a FLOSS BSD UNIX based operating system developed by Berkeley University of California back in 1993.

It is widely used, and it is possible to categorize three kind of consumers for this operating system:

- Large organizations offering network and telecommunication services, such as Internet Service Providers
- Hardware manufacturers of appliances or products with embedded operating systems, such as computers or firewalls
- Personal computers and the way that everybody access to the internet nowadays, in the sense that software libraries and protocols from FreeBSD are needed for this purpose

1.1.2 Main goals of FreeBSD

- Maintain an efficient social process of writing software Based on the interaction and feedback of the entities involved
- Develop a robust, secure and reliable operating system With about 340 CVS committers and thousands of contributors involved in the project
- Large documentation available, including translation for multiple languages
- Maintain developing under Revision Control System At using Apache Subversion (SVN)
- Ensure freedom of licensing Thanks to the permissive FreeBSD License

- Support the project through the FreeBSD Foundation By providing financial resources, legal advice, receive donations that are destined to help developers get to conferences and events, as well as research investment and hardware purchases
- Well define the role of the people involved in the project, depending on their perspective (ISP, developer, final user, etc.)

1.2 How Open Source Projects Survive Poisonous People

A talk given at Google's Open Source Developer Speaker Series by Ben Collings & Brian Fitzpatrick, who are in charge of the project Hosting Feature on Google Code Site by January 2007, in order to discuss:

- A possible way to run a free open source software community
- Understand the community as the main resource to focus the attention
- Guidelines proposed by Karl Fogel in the book "Producing Open Source Software" [Fogel, 2002]
- Try to reach a culture of trust in the community

1.2.1 Brief description of what is meant by poisonous people at FLOSS communities

A software development community is constantly besieged by external or even internal factors that could fractured it, even more in a FLOSS community. Factors that can be attributed to people as well as procedures performed daily. It is extremely important to identify these elements and know both the attackers and vulnerabilities, in order to achieve sustainability of the Open Source project.

1.2.2 Main goals in Open Source Projects to survive poisonous people

It is possible to identify four stages of protection, and specific goals for each of them are the following:

- Comprehension:
  - Understand people to protect from
  - Start with coding, avoid paralysis, do not be that perfectionist
  - Ignore noise minorities without constructive arguments

- Fortification:
  - Fortify the project against poisoned people (the threat)
  - Keep the community resistant to infection
  - Maintain politeness, respect, trust and humility. The healthy of the community depends on it
  - Promote best practices, code-collaboration policies
  - Have well defined processes
  - Set goals when coding, then limit the scope, avoiding limitless features loop
  - Rely on mailing lists, the main communication method in any FLOSS project
  - Do no let people open old discussions, unless it is something that really worth it
  - Produce documentation on mission and objectives, bug fixes, issue trackers, so new developers are able to continue with the guidelines without repeat mistakes
  - Define standard log formats to follow it in an easy way
  - Keep coordination with commiters about changes
  - Do not allow commits of very large changes on the project's code, try to be incremental (using branches)
  - Identify the Bus Factor of the project: the number of developers that have to get hit by a bus to leave the project on chaos
  - Encourage people to spread their expertise around
  - Reduce territoriality
  - Promote equal values about the project among committers
  - Consider voting/veto/forks when necessary in pro of the community (as a last resource). It is healthier try to come to resolutions on relatively easy basis
- Identification:
  - Look or notice for people who really help the project to evolve
  - Encourage people to implement NETIQUETTE
  - Do not let others to blackmail the project on groundless basis
  - Keep the values over certain new proposals (like new unnecessary features)

  - Give more importance to the coding design over the code
  - Discuss features with the community

- Disinfection:
  - Look for pushing out the people disrupting attention and focus of the project
  - Always asses the damage of situations and decisions
  - Pay attention on newcomers, letting them a chance
  - Look for the facts over the emotions

1.3 SilverStripe CMS

This talk was held in August 2007 by two of the founding members of the project, Sigurd Magnusson and Sam Benny. A pair of FLOSS enthusiasts from New Zealand, who have participated in the Google Summer of Code competition. This couple claims that FLOSS is a democratic process in New Zealand and that this system is in fact reproducible.

1.3.1 Brief description of SilverStripe Content Management System

Magnusson and Benny started building websites since year 2000, and in this process they came to accumulate deep knowledge about the tasks involved in this business. They developed an application that describe as:

- Bigger than a blog tool
- More nimble than existent open or closed source CMS enterprise systems
- Focus on usability for website managers
- Out of the box, and at the same time providing a framework

1.3.2 Main goals of SilverStripe

- Use FLOSS to advertise the product and get people on board, with no ads more than the own source code
- Give a chance to FLOSS business model, as they offered the product as proprietary at first with no luck
- Rely on FLOSS as a cheaper and more effective way to market

- Use all the lessons learned and make it available to potential many other contributors
- After having based their infrastructure on FLOSS, give back something in return to communities
- Establish guidelines when managing people on a FLOSS project
  - Establish a road-map
  - Give public recognition
  - Motivate the community in a proper way
  - Delegate work when necessary
  - Encourage group answers in reply to questions from individual users, as everybody is noticed
  - Try to mentor herding (guidance in the process)
- Document mistakes and their replication, so other users/developers do not commit them again
- Participate in events such as Google Summer of Code due to Mash-ups, functionality integration from third parties is real fun and enriching
- Internationalization is an important factor to disseminate
- Provide support over tools, by services and hosting for instance
- Promote web conferences, face to face meetings are healthy and necessary
- Look for diversification of the project
- Predict what people want from the product

1.4 Camino

Talk held in January 2007 by Mike Pinkerton, a software engineer in MacDev at Google, who worked at Netscape from 1997 until 2002. Pinkerton tries to highlight the lessons learned in the process of develop a FLOSS project around a web browser, in a time that were only two direct competitors “Netscape Communicator” and “Microsoft Internet Explorer” back in 1998.

1.4.1 Brief description of Camino

It started as a Mozilla initiative to continuing developing a web browser for the Mac platform, due to cross-platform was an important feature for the browser market, and it still is at these days. But in the very beginning this project was born as business strategy by Netscape, who created a business model around selling their product: a commercial web browser. The main problem faced by this company was that Microsoft started giving it for free as part of Windows OS. Anyhow Netscape realized that Microsoft's strategy could be used against them in a FLOSS field, because Internet Explorer was highly integrated with its also proprietary OS. By turning the Netscape Communicator source code available, they would be competing in the opposite direction to Microsoft's strategy, to whom opening the source code was non-viable. This is how in March 1998 was announced the release of Netscape Communicator source code as the Source 331 Project.

1.4.2 Main goals of Camino as a FLOSS project

- Compete against Microsoft or defeat it in the web browsers field
- Trust in the support of the community. The lasts entire release cycles of Camino were based on the open source community, with no Netscape support at all. The quality for Camino Beta releases was very high, they didn't want to let the community down. There was a documentation group, testers, designers, bugs reviewers, etc.
- Defend values. Netscape crew had to wear two hats, one for the Corporation continuing with Netscape developing, and on the other hand with Mozilla's community
- Accept changes when required. Raptor project was another rendering engine with better performance at the time. Also it was difficult to continue the relationship with Mozilla Foundation at certain point
- Respect the community. Management decision about change the direction of the project was not properly announced, and consequently much of the credibility on the project was lost. Later when Camino project split apart, started to take Mac community more seriously
- Understand the different roles in the community. For instance there are two distinct groups, the designers (following standards), and the programmers (implementing

the requirements), try to achieve a balance or reduce the gap between them is crucial

- Seek evolution. Since 2000 developers started to work into improve performance, reduce memory consumption, incorporate a new rendering engine. The new develop aims to be native, fast, keep compatibility (including Cocoa API's, Raptor and Gecko)
- Keep the community motivated. Camino was one of the last effort products from Netscape for the community. Improvements such as an user friendly release and attractive website at the time of 1.0 mayor version, produced 200 downloads per day
- As Eric Raymond's Cathedral and Bazaar [Raymond E., 1999]: be open as the promiscuity, share as much as you can to build a really committed community, involve users and developers
- Camino did not followed the "Release Early, Release Often" directive, with the intention of maintain high quality before releasing
- Mozilla.org was Developer-Centric (forgetting the users), but Camino followed another direction
- Camino looked for reach end-user developers, offering forums and Q&A, having the right focus on the website
- Modules ownership is not a good practice according to Camino project, because it centers the whole development process in individuals, nevertheless it is sometimes necessary to avoid dispersion. A weak owner is worst than no owner at all
- Consider the testers as the most valuable resource. For instance, they just want a functional browser no matter what engine it has, and they will do all necessary tests to obtain it
- It is imperative to have an Open Bug Database to file code issues
- Understand that it is impossible to please everyone, it educates the community

1.5 Greg Kroah Hartman on the Linux Kernel

In 2008 Greg Kroah Hartman, as a subsystem kernel maintainer for USB, driver core, and previously PCI related, gave this talk about the Linux Kernel, the process, who is doing it, how it is been done.

1.5.1 Brief description of the Linux Kernel ecosystem

The 1.0 version of the Linux Kernel was released on March 1994. At this time it only supported single-processor i386-based computer systems, as well as a reduced set of hardware drivers in general. In subsequent releases, portability became a concern, so the kernel gained support for computer systems using processors based on the Alpha, SPARC, and MIPS architectures, and also began to incorporate drivers to support a large number of hardware devices. This evolution was possible because many hackers became interested in the project proposed by Linus Torvalds who licensed it as GNU GLP v2 [Torvalds L., 1991], which allowed and facilitated the rules for sharing the source code since the very beginning.

1.5.2 Main goals of the Linux Kernel project

- Provide a stable-multiplatform operating system kernel
- Encourage to include code from the community into the kernel
- Keep the current development scheme without following an strict hierarchy, as it is still organized enough and efficient. Linux is not intelligent design, is evolution
- Support for new features is a day to day work
- Testing is about running the kernel and check if it still works, rely on developers and users of the community
- Encourage companies to increasingly sponsor more Linux Kernel developers
- Lead questions like if it is a good idea to let current work-founding companies impose the direction of Linux Kernel development
- “Regression” is a way to correct mistakes
- Enterprise boxes (such as kernel stable releases) are not something Linux Kernel Community maintain, it is a matter of the companies supporting it

- Currently making emphasis on KVM, and the possibilities to control real time costs (CPU usage, execution time, network resources, etc.)

1.6 Mercurial Project

Continuing with the Google TechTalks series, in June 2006 Bryan O'Sullivan talks about a new project oriented to distributed revision control systems. Mercurial was started as an appropriate tool likely to FLOSS developments, allowing the evolution of such projects.

1.6.1 Brief description of Mercurial

Written in Python an C programming language, under GNU GPL v2 license, Mercurial is a cross-platform, distributed revision control tool for software developers. This project started as an alternative to Version Control Systems (VCS) such as BitKeeper, who was responsible until those days for holding the development scheme of important and robust software projects such as the Linux Kernel. Although it was efficient, communities like the one from the Linux Kernel were determined to change their proprietary VCS to a better and flexible solution, so new initiatives were born in parallel and by different people (mainly BitMover and Git) [Mackall M., 2006].

1.6.2 Main goals of the Mercurial project

- Produce a reliable and sustainable Revision Control System
- Keep written in Python, be distributed, run fast with good performance, and be easy to understand, so developers can focus on their own projects
- Mercurial's major design goals include high performance and scalability, decentralized, fully distributed collaborative development, robust handling of both plain text and binary files, and advanced branching and merging capabilities, while remaining conceptually simple
- Displace competition as Subversion, Bitkeeper or Git
- Maintain a business model around cloud storage plans
- Invest efforts in efficient patch handling

1.7 Stefano Zacchiroli on Debian: 20 Years and counting

This time the current Debian Project Leader Stefano Zacchiroli by year 2013 talks about the success of the Debian project as a FLOSS community and the factors that have allowed it over the last 20 years.

1.7.1 Brief description of the Debian project

Debian itself is a FLOSS operating system formed mainly by software under GNU General Public License. It was founded by Ian Murdock in 1993, and has been continued to develop since then by a community of people who collaborate as volunteers following the Debian Free Software Guidelines (DFSG) [Perens et al., 2004]. The software, the community, and the set of rules under which they coexist, is what gives life to the project as a whole.

1.7.2 Main goals of Debian as a project

- States the importance of the Debian Freedom Manifest
- Focus on developers and users
- The first idea was to turn GNU/Linux competitive to existent commercial OS's
- Provide an easy method to install
- Build software in a collaboratively way by experts of many different areas
- Be the first mayor distro developed openly in the spirit of GNU
- Count with the support of the Free Software Foundation
- Innovate at introducing GNU/Linux distributions in a collaboratively way
- An OS meant to last on time, installable on Servers and Desktops
- 1/3 Debian: the OS
  - Produce stable product versions looking for binary distributions, completely free, released every 24 months, supporting many architectures, provide repository archiving and security support (3 years)
- 1/3 Debian: The Project

  - Common Main Goal: Create the best possible Free OS. Based on the following documents
  - Debian Social Contract (1997): agreement between no only developers, but among the whole ecosystem
    . Releases 100% FLOSS
    . Give back contributions to community
    . Do not hide problems
    . Establish priorities: users and FLOSS
  - Debian Constitution (1998):
    . Define the rules of a Free Software-compatible democracy
    . Being a strong motivation for people who join to the project

- 1/3 Debian: The Community
  - Transparency as a principle
  - Provide empowerment to members of the community. Based on the public nature of the source code and the impact that any participant is able to produce with the proper skills
  - Maintain high volumes of communications: mailing lists, IRC, web services (social network principles-compatible like identi.ca)

- As the project evolved, the goals aimed at:
  - Produce a faster and robust dependency-based boot system
  - Provide a completely free Linux Kernel, excluding proprietary firmware
  - Offer a large choice of pure blends or customized distributions (education, medicine, science, etc.)
  - Rely on online services supporting user demands

- Nowadays:
  - Maintain a multi-architecture (cross-compilation, package share among architectures, mixed 32/64 bits binaries)
  - Include Private Cloud Support (OpenStack, XEN) and Public Cloud Support (EC2, Azure)
  - Handle more end-user desktops (Gnome, KDE, Plasma, XFCE)
  - Guarantee easy system upgrades

  - Support new architectures: armhf (such as handheld devices), s390x (virtualization servers)

- Debian Package Quality:
  - Keep the culture of technical excellence
  - Continue the package design, testing and maintenance scheme
  - Release Mantra: "We release when is ready"
- Attachment to freedom
- Independence (not commercial companies behind)
- Defined decision making rules:
  - Do-ocracy, where individual developers are able to make any decision regarding their own work
  - Democracy through a voting system
  - Consequences:
    - Reputation follows work
    - No benevolent dictator, button-up scheme
    - No imposed decisions from power figures (such as corporations)
- Recently there are many derivative distributions that depend on Debian, and this is a way to reach new potential users/contributors (everybody wins)
- The intention is to guarantee the sustainability of the Free Software Ecosystem
  - Return in back (reduce patch flow viscosity)
  - Give credit when apply
- To contribute, there are standards to accomplish
  - It is not about provide a package and leave
- Receive punctual contributions by monetary donations, intended for resources, face to face meetings, conferences, etc.
- Promote work-contribution:
  - End-users: testing, reporting, fixing bugs, monitoring packages

  - Developers: packaging new contributions
  - Non-development tasks: translation, themes design, media diffusion, documentation, events, etc.

- Make it clear that join Debian imply a commitment

- A recent Diversity Statement was recently published to help increase the quality of the community, adopting all kind of contribution and contributors meeting the Debian Freedom Manifest

1.8 Debian Secrets what I wish I knew before joining Debian

This talk was given at the Free and Open Source Development European Meeting (FOSDEM) in 2013 by Lucas Nussbaum, the current Debian Project Leader. This is an European event centered around FLOSS development, aim to enable developers to meet and to promote the awareness and use this kind of software and form or style of development. Nussbaum discuss about some well defined patterns in the Debian community in its current status.

1.8.1 Brief description of some patterns identified in Debian project

As a 20 year old project, people from Debian's community have identified some typical and recurring behaviors that do not help the overall strengthening of the project. Figures such as the project leader are aware of this phenomenon, and make it public with the purpose that the members of the community could be able to avoid them. Factors ranging from the rules governing the project, people involved and their motivation, to development models that are followed, revealing the inner workings of Debian.

1.8.2 Main goals of Debian community to identify patterns

- The establishment of a do-ocracy ruled by a technical committee
  - In general maintaining a culture of technical excellence
  - Should not overrules the maintainer of a package
  - Is a long process, could take months to make a decision
  - Rarely goes to voting, considered as a last resource
  - Core teams take care of key areas of the project, taking good decisions most of the time

  - When not, it results in flamewars, hurting the project badly

- The presence of Benevolent Dictators
  - Eventually there exists some territoriality in certain Debian areas

- File a bug and send a patch philosophy
  - In general for small bugs, it is not efficient

- Nobody feels empowered to do the work
  - No strict hierarchy, possible blocking decisions by not knowing the leadership
  - Encourage to do it, anyway it can be reversed

- Nobody wants to do the grunt work
  - Work in terms of other areas but computers. It is recently changing (communication, accounting, marketing, etc.)

- Over optimized processes
  - Some developers do it for themselves, but for many it is possible to rely on Bug Tracking Systems

- Button-up designs and adoption
  - Debian follows a decentralized development model
  - This process generally starts by trying to solve own problems, then expands to other teams, finally to the whole project. It is a long process, eventually it takes months or years
  - Package Tracking Systems (PTS), a Centralized Source Package Dashboard, turns increasingly imperative due to the diversification of the project
  - Developer Packages Overview (DDPO), a Centralized Maintainer Info, due to overwhelming number of contributions

- It is necessary to keep looking at changes over time
  - snapshot.debian.org provides a list archive for packages in previous Debian releases

References

[Fogel, 2002] Fogel, Karl. “Producing Open Source Software: How to Run a Successful Free Software Project”. O’Reilly Media, 2005. p. 1-202.

[Mackall M., 2006] Mackall, M. (2006). “Towards a Better SCM: Revlog and Mercurial”. Linux Symposium Proceedings. Ottawa: Selenic.

[Perens et al., 2004] Perens, Bruce, et al. “Debian Free Software Guidelines”. Debian community. Version 1.1, 2004.

[Raymond E., 1999] Raymond, Eric S. (1999). The Cathedral & the Bazaar. O’Reilly, pp.279.

[Torvalds L., 1991] Torvalds, Linus. (1991). Linux Kernel GNU GPL License v2: “Linux Kernel Copying”. DOI=http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/COPYING