Case Studies II - Community Talks

Technologies and Contributions

Ricardo García Fernández

April 3, 2013

Contents

1 Introduction

Through this document I want to summarize relation between technologies and contributions in those FLOSS project talks from Subject Case Studies II at URJC. A brief introduction to them and a final comparison table.

2 Apache Software Foundation

Apache Software Foundation - **ASF** - 501.3 epigraph C nonprofit Foundation

ASF defines itself as:

> *"not simply a group of projects sharing a server, but rather a community of developers and users"*

A *Foundation* is a legal umbrella to allow us to legality providing transparency.

2.1 Technologies

Apache could be defined as a community of communities. Therefore contributors enter for each community through its rules. The basic rules are inherited and common from ASF.

To make a contribution, first, the contributor must be in contact with the particular community and its ecosystem project.pse, EUPL, FLOSS, FLOSSIE, flos

In Apache projects bear a similarity in ecosystems, because they have been through an acceptance process to be adopted or published by Apache.

As each project has:

- *Mailing lists.*
- *Source code.*
- *Bug tracking.*
- *Mentors.*
- *Committers.*
- *Documentation* (wikis, html, pdf...).

Therefore, in order to help you to follow the guidelines set. It is recommended in all FLOSS projects you can register for mailing lists as a first step to find out how it works and basic communication. Then begin to understand the design, installation, use and testing.

You can read a summary of the basic rules for contributors http://www.apache.org/dev/contributors.html ASF.

To become a contributor, you have to be invited by the other contributors of the project. Having won merits and respect within the community.

2.2 How to Contribute

If you want to become a contributor in a project at Apache Software Foundation, a tip, you could start following incubator projects.

Figure 1: Apache Incubator

For a project, this is the previous step before become an Apache project. These projects are looking for create a hudge community and it's easier to become a contributor than in bigger projects from ASF.

Every new committer must read Apache Software Foundation Guide introducing all aspects that are needed for a committer:

- *Contributor License Agreement* - to contribute with your commits to ASF.
- *Account creation* - step by step using ASF standards.
- *Responsibilities* - as a member of ASF, personal web space, community, ApacheCon.

You can find an extense explanation in ASF new committers guide.

2.3 Committers resources

An interesting link of how decisions are made in the Apache Software Foundations to see meritocracy in action:

- `http://community.apache.org/committers/`

3 WebKit

WebKit is a layout engine software designed to allow web browsers to render html (is not only on web pages). Become a FLOSS project in 2005 when Apple released the code.

Is used in Mac OS X system framework with Safari, Dashboard, Mail, and many other OS X applications. Also you can find WebKit in Nintendo 3Ds, Kindle and Android.

This project started as a branch of the KHTML and KJS libraries from KDE. It is a community made up by companies: Apple, Nokia, Google, RIM, Igalia, Samsung. Because most contributions are made by companies, I want to emphasize that the company that provides more code to WebKit is Google.It is strange to see this information on a project controlled by Apple.

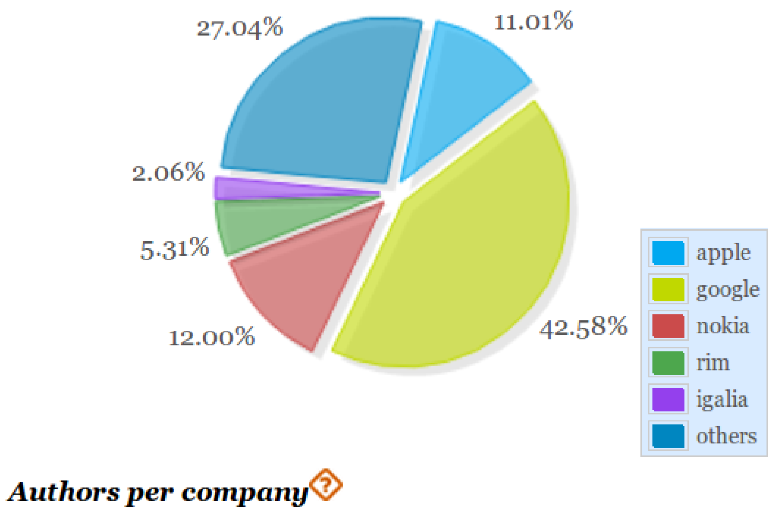


Figure 2: Bitergia WebKit analysis: `http://bitergia.com/public/reports/webkit/2013_01/`

You can see better numbers in the study by Bitergia on WebKit community posted on February 6, 2013, Bitergia blog.

3.1 Technologies

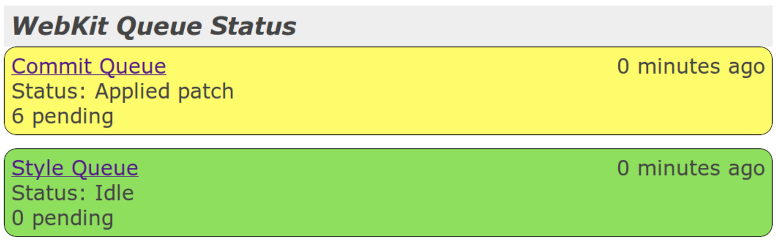
Documentation :The technologies surrounding the WebKit project, which we consider as ALM Tools, (Application Lifecycle Management):

- Bug Traking System (BTS) Bugzilla - `https://bugs.webkit.org/`

- Issue Tracking System (ITS) Trac - `http://trac.webkit.org/`
- Early Warning Systems (EWS) Bugzilla - `http://webkit-commit-queue.appspot.com/`
- Developer Guides for each Operative System - `http://www.webkit.org/building/tools.html`

I will highlight as the **EWS** tool. *EWS* offers a tracking code contribution we have made to the project (via a patch). It reflects the results of each test in each type of platform. WebKit looks much the appearance of the test because a change can affect millions of users.

As we can see, the system shows a first image of the status of contributions.

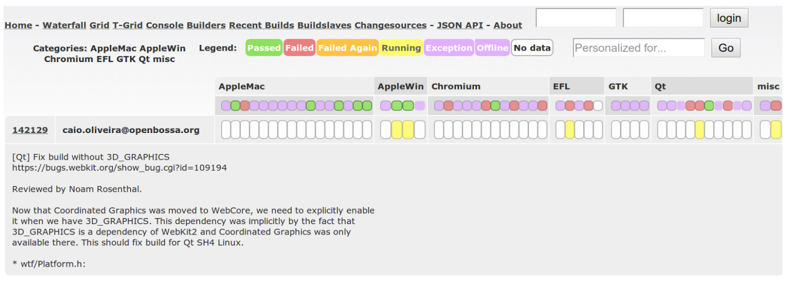

WebKit Queue Status

Commit Queue 0 minutes ago
Status: Applied patch
6 pending

Style Queue 0 minutes ago
Status: Idle
0 pending

Figure 3: Queue status

Accessing BuildBot WebKit Console, we have access to the results and follow-up of each of the patches and tests for each of the existing distributions (ports).


Home - Waterfall Grid T-Grid Console Builders Recent Builds Buildslaves Changesources - JSON API - About
login
Categories: AppleMac AppleWin Chromium EFL GTK Qt misc
Legend: Passed Failed Failed Again Running Exception Offline No data
Personalized for...
Go
AppleMac
AppleWin
Chromium
EFL
GTK
Qt
misc
142129 caio.oliveira@openbossa.org

[Qt] Fix build without 3D_GRAPHICS
https://bugs.webkit.org/show_bug.cgi?id=109194

Reviewed by Noam Rosenthal.

Now that Coordinated Graphics was moved to WebCore, we need to explicitly enable it when we have 3D_GRAPHICS. This dependency was implicitly by the fact that 3D_GRAPHICS is a dependency of WebKit2 and Coordinated Graphics was only available there. This should fix build for Qt SH4 Linux.

* wtf/Platform.h:

Figure 4: Webkit build bot

It is very rigorous system of contributions to the project, we will see in the next section.

WebKit has a very useful development guidelines documentation:

- *Coding style guidelines* - `http://www.webkit.org/coding/coding-style.html`
- *Technical Articles* - `http://www.webkit.org/coding/technical-articles.html`
- *Regression Testing* - `http://www.webkit.org/quality/testing.html`

- *Reporting Bugs* - `http://www.webkit.org/quality/reporting.html`
- *Bug Life Cycle* - `http://www.webkit.org/quality/lifecycle.html`
- *Wiki Documentation* - `http://trac.webkit.org/wiki`

3.2 How to Contribute

In WebKit there are many ways of contributions; translations, code, documentation, etc. We are going to focus on contribute with code.

There is a guideline explaining how to contribute code. These are the steps you have to follow to contribute code in WebKit:

- Choose or create a bug report to work on.
- Develop your changes.
- Make sure your changes meet the code style guidelines. The check-webkit-style script may be of help.
- Run the layout tests using the run-webkit-tests script and make sure they all pass. See the testing page for more information, as well as what you need to do if you've modified JavaScriptCore.
- Add any new files to your working directory.
- Prepare a change log entry. You may have to add entries to multiple ChangeLogs. The prepare-ChangeLog script will create stub entries for you. See the paragraph about ChangeLogs below.
- Create the patch using the svn-create-patch script.
- Submit your patch for review to bugs.webkit.org.
- Make any changes recommended by the reviewer.
- Once reviewed, ask someone to land your patch or mark it for automated commit.
- Please watch for any regressions it may have caused (hopefully none)!

3.2.1 Committers and Reviewers

WebKit contributors are divided into two groups; *committers* and *reviewers*. The committers can contribute code to the reviewers agree.

To be a committer, you have to be nominated by the reviewers. Having a minimum number of significant patches *(ranges between 10 and 20)*. Bureaucratic process and receive approval of 3. Is searching the plurality of contacting between one or more reviewers.

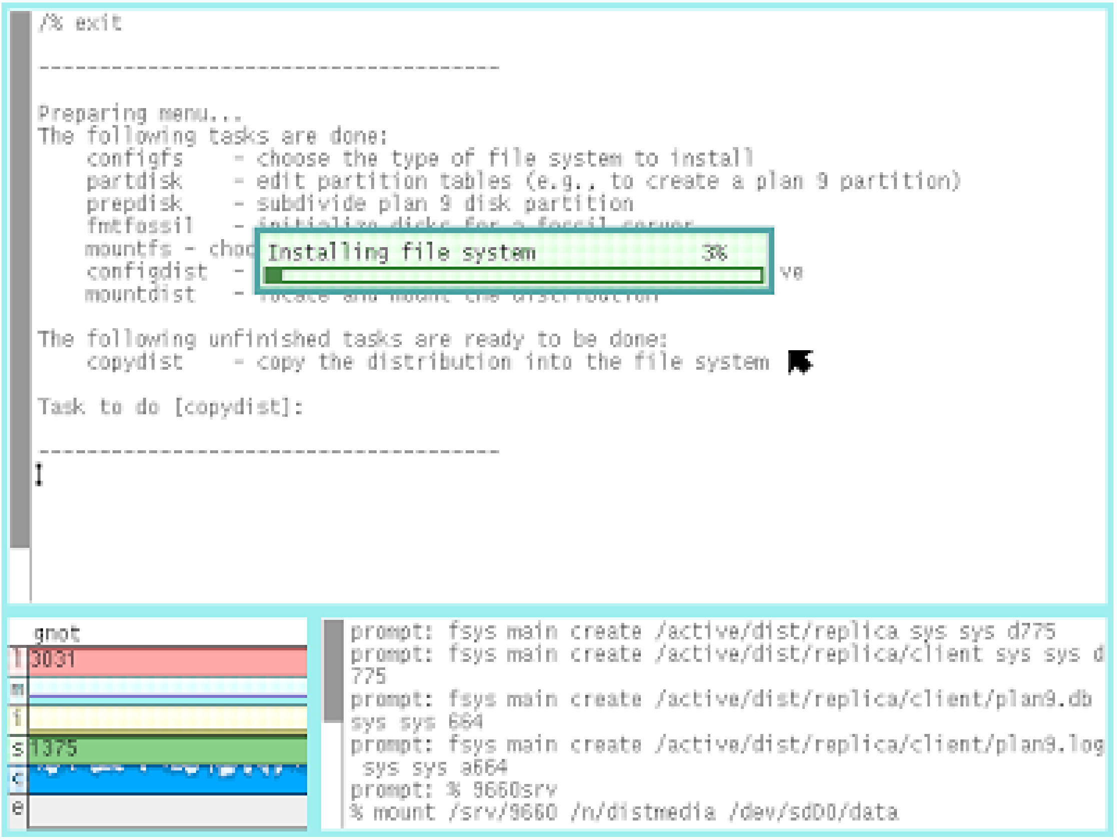
> *Any company with more than three reviewers may give permission to a reviewer.*

Becoming reviewer, is a higher jump. You have been elected by *reviewers from different companies and have contributed over 80 functional patches.*

Both steps are accompanied by the signing of an agreement with Apple, not with the WebKit project.

4 Plan 9

What is Plan9 ? A movie from Ed Wood ? A 'new' Operative System ?*Plan9 from Bell Labs* is an Operative System developed between the 80's and 2002 by the research and development subsidiary of the French-owned Alcatel-Lucent in Berkeley Heights, New Jersey, United States.*Yes, the name is from the movie :)*



```
/% exit

--------------------------------------

Preparing menu...
The following tasks are done:
    configfs    - choose the type of file system to install
    partdisk    - edit partition tables (e.g., to create a plan 9 partition)
    prepdisk    - subdivide plan 9 disk partition
    fmtfossil   - initialize disks for a fossil server
    mountfs - choo
    configdist  -                                             ve
    mountdist   - locate and mount the distribution

The following unfinished tasks are ready to be done:
    copydist    - copy the distribution into the file system

Task to do [copydist]:

--------------------------------------
```

```
Installing file system                3%
```

```
prompt: fsys main create /active/dist/replica sys sys d775
prompt: fsys main create /active/dist/replica/client sys sys d
775
prompt: fsys main create /active/dist/replica/client/plan9.db
sys sys 664
prompt: fsys main create /active/dist/replica/client/plan9.log
 sys sys a664
prompt: % 9660srv
% mount /srv/9660 /n/distmedia /dev/sdD0/data
```

Figure 5: Plan9 talk by *Enrique Soriano Salvador*

Plan9 was developed by the same developers of Unix to create a real network OS where *everything* is a file. Even the processes. Everything is related to the basic input/output file readers, the main purpose in the ancient Unix.

4.1 Technologies

*Plan9*is developed using C language. The tools they use to collaborate are:

- *Mail lists* - to get in touch with the community and contribute tool `http://plan9.bell-labs.com/wiki/plan9/mailing_lists`.

- *patch* - Command (*patch*)to create file changes.
- *Online Sources* -`http://plan9.bell-labs.com/sources/`
- *TODO* -`http://plan9.bell-labs.com/wiki/plan9/TODO/index.html`
- *Contrib index* -`http://plan9.bell-labs.com/wiki/plan9/Contrib_index/index.html`
- *Irc channel* -`http://plan9.bell-labs.com/wiki/plan9/IRC/index.html`
- *Workshops* - `http://www.iwp9.org/`

4.2 How to Contribute

Contributing to the community is difficult, because all your contributions have to pass Plan9 filter handled by one person. You don't depend on a community itself.

By other way, you have a very good guidelines and references by very good C developers to develop Plan9.

- don't use // comments; some old Plan 9 code does, but we're converting it as we touch it. We do sometimes use // to comment-out a few lines of code.
- avoid gotos.
- no tabs expanded to spaces.
- no white space before opening braces.
- no white space after the word "if", "for", "while", etc.
- no braces around single-line blocks (e.g., if, for, and while bodies).
- integer-valued functions return -1 on error, 0 or positive on success.
- functions that return errors should set errstr(2).
- variable and function names are all lowercase, with no underscores.
- enum or #defined constants should be Uppercase or UPPERCASE.
- struct tags are Uppercase, with matching typedefs.
- automatic variables (local variables inside a function) are never initialized at declaration.
- follow the standard idioms: use x<0 not 0>x, etc.

The end they follow a very common development filosophy:

> *Ultimately, the goal is to write code that fits in with the other code around it and the system as a whole. If the file you are editing already deviates from*

> *these guidelines, do what it does. After you edit a file, a reader should not be able to tell just from coding style which parts you worked on.*

In my opinion, *Plan9 is a very good Operative System to learn how an Operative System works and enjoy learning.*

5 Mozilla

Mozilla foundation is a huge community that involves different types of users and has a central message applied to all its levels:

"Our mission is to promote openness, innovation & opportunity on the Web." In Mozilla Manifesto you can read the whole document about freedom and accesibility that they apply day by day.

The Foundation is consolidated by years of dedication to freedom of the web through its flagship Firefox. Since Netscape released the source code in 1998 Mozilla got down to work to develop a web browser from FLOSS source code they received from Netscape.

Mozilla wanted a free internet because IExplorer market share was 90%. Was a private market and now this trend has changed. Source: StatCounter Global Stats - Browser Market Share

5.1 Technologies

There is a bunch of technologies surrounding Mozilla development communities and of course, projects:

- *Firefox* - Web Browser with a particularity that doesn't have portability to IOS because it uses Gecko instead of Webkit to render content. Apple doesn't allow a browser (or anyhting) that is not reder usgin Webkit.
- *FirefoxOS* - New Mobile operative system based in HTML5 and Browser Oriented.
- *Thunderbird* - eMail management but abandoned project. Developers have changed to Mozilla.
- *Camino* - MacOS Mozilla Browser using Webkit
- *Seamonkey* - Firefox, Thunderbird, Chat, Web Editor.
- *Sundbird/Lightning* - Calendar project from Mozilla.
- *XULRunner* - It's like Java Virtual Machine allowing multiplatform products oriented to Mozilla packages.

And of course, developer tools to work with Mozilla community:

- *Mailing lists*- For developers, information, everything related to Mozilla.
- *Bug Tracking System*-Bugzillafor every product.
- *Source Code Managemet*- Baazar, Mercurial, SVN, CVS.
- *MXR*- Cross references about the code from Mozilla (mxr).
- *Forge for scm*- GitHub.

- *QMO*-Home of Mozilla's testing andquality assurance community.
- *Crash Stats* - *Continuous integration* server output visualizations usingSocorro.
- *Graph CI*- Graphic visualization for every built from the code ingraph section.
- *Tinderbox*- Tool to navigate throughbuild logs and results.

5.2 How to contribute

Mozilla community is very community, this statement may sound redundant, but would be an accurate definition. He moved away from the technology and developers as standard for any project.

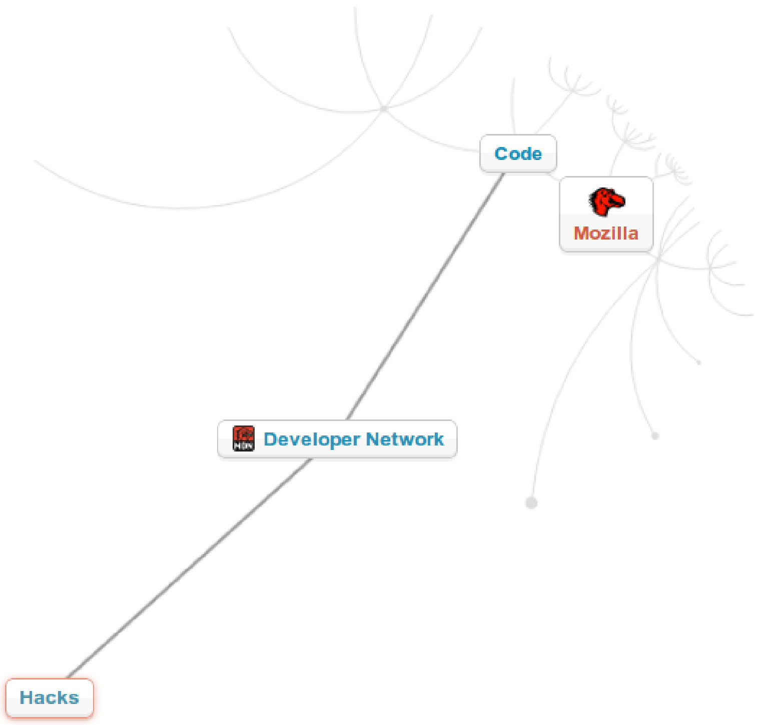


Figure 6:

Mozilla has the basic steps as other FLOSS projects for How to Contribute and gives you an opportunity to contribute in more than one way looking for a chance:

- Help for Users
- Quality control
- Programming

- Spread the word
- location
- Web Development
- Accessories
- graphic Design
- Documentation and drafting
- Education

Furthermore gives you the opportunity to look for contribution near your location, a very useful tool. Mozilla invites you to be part of an Open Web. You don't have to be the smartest guy in the world they do this work spreading that all help is good an they will find for you the best place to help *everyone*with your contribution.

6 GNU

Starting with the acronym inception definition of GNU *"GNU is not Unix!"* in 1983. GNU is divided in three departments:

- *The GNU Project.*
- *GNU Software* - Develops GNU software.
- *The GNU System* - GNU Software + external Software.

GNU selected Unix because of simplicity instead of VMS. Composed by different independent utilities. Decoupled and modular, so the choice to clone the system.

GNU Project is a Global Project, hasnt legal entity. RMS (Richard Matthew Stallman) created *Free Software Foundation* (FSF) to convert legal resources to GNU hackers.

6.1 Techonologies

All projects inside GNU universe share a homogeneous structure and tools to contribute. A part from common techonoligies they use specific tools for each project, depending of the tool purpose.

Here is a common list for every GNU project:

- Webpage
- Development
- Manual
- Reporting bugs
- Getting help

We apply this structure to an existing GNU project: GNU Recutilsand see how easy is to get all starter information.

GNU Recutils is a set of tools and libraries to access human-editable, plain text databases called recfiles.

- Webpage -`http://www.gnu.org/software/recutils/`http://www.gnu.org/software/recutils/
- Development -`http://savannah.gnu.org/projects/recutils/`http://savannah.gnu.org/projects/r
- Manual -`http://www.gnu.org/software/recutils/manual/`http://www.gnu.org/software/recutils/
- Reporting bugs -`http://savannah.gnu.org/bugs/?group=recutils`http://savannah.gnu.org/bugs/
- Getting help - FAQ`http://www.gnu.org/software/recutils/faq.html`http://www.gnu.org/softwa

- Mailing lists - bugs`http://lists.gnu.org/mailman/listinfo/bug-recutils` and help`http://lists.gnu.org/mailman/listinfo/help-recutils`.

You have total freedom from the moment when develop, test, and evolve Recutils from tools or guides that offer.

At last but not least important: Following GNU coding standards that are very important to develop readable code.

6.2 How to Contribute

At this section you will wonder, how I can contribute in a GNU project? one of the most important projects in history?It's easier than it at first seems, of course, following some basic rules of coordination and participation.

In an overall picture youcan recognize this structure:

- *RMS* - Appoints maintainers. *"GNUism"*.
- *GNU Advisory Committe* (2009) - Advisory, communication, transversal vision, external contact. advisory@gnu.org. 8 people, most maintainers and FSF America and Europe vice-presidents.
- *Maintainers* - Maintainer, lead program development. Follow GNU policies for maintainers.
- *Contributors* - Developers with or without commit access.

You start in the last point as a contributor with ot without commit access, because GNU is not only writting code. There are many areas: documentation, translations, blogging, law, in which where you can be more helpfull that you think. Take a tour in How to help GNU.

But, focussing strictly on development you start as a non commit contributor, inside a mailing list following basic guides. Explaining Recutils project guidelines:

- *Test releases* - Trying the latest test release (when available) is always appreciated. Test releases of Recutils can be found at http://alpha.gnu.org/gnu/recutils/ (via HTTP) and ftp://alpha.gnu.org/gnu/recutils/ (via FTP).
- *Development* - For development sources, bug and patch trackers, and other information, please see the Recutils project page at savannah.gnu.org.
- *Translating Recutils* - To translate Recutils's messages into other languages, please see the Translation Project page for Recutils. If you have a new translation of the message strings, or updates to the existing strings, please have the changes made in this repository. Only translations from this site will be incorporated into Recutils. For more information, see the Translation Project.

- *Maintainer* - Recutils is currently being maintained by Jose E. Marchesi. Please use the mailing lists for contact.

Starting from these test releases to get an inmersion in the project and know its architecture. This starting point is very important for FLOSS project because is an starting mentoring guide through the code and project structure.

After this step you can know how to read (better not perfectly) the code, understand bugs, follow solutions and develop your first patch. Here is your gateway to contribute with code to a GNU Project and the path to become a maintainer, only if you want to learn (for 'gratis').

Other ways to contribute, as I explained before, are translate projects and develop documentation. José E. Marchesithat explains what is and what gives you 'Software Libre': Bicicletas y Software Libre.

> *Ayer me compré una bicicleta. Y es algo fantástico! Puedo utilizarla tanto en la ciudad como en el campo. Y en cualquier mes del ao! Puedo darme paseos y, si algún día me hace falta, puedo repartir periódicos y sacarme unos durillos. Mi bici no traía luces, pero no es problema: le he puesto una dinamo y una buena luz, incluso lo he hecho yo mismo: ahorras y aprendes al mismo tiempo! Además la hemos pagado a medias entre mi compaera y yo. No es ningún problema compartirla. Pero quizá algún día nos cansemos y queramos librarnos de ella. Podremos venderla de segunda mano. O se la regalaremos a alguien que la vaya a usar.*
> *Hoy me he comprado un programa de ordenador. No es nada muy sofisticado (un cliente para leer el correo), pero tiene buena pinta y hace mogollón de cosas. Me ha salido bastante caro porque no he podido comprarlo a medias. Me dicen que ella no puede utilizarlo, aunque lo necesita. Además sólo podré utilizarlo durante un ao. Me han dicho que pasado ese tiempo debo pagarlo otra vez y usarlo durante un ao más y no puedo engaarles: el programa se rompe y ya no funciona más. Curiosamente he leído que si utilizo este programa en Corea del Norte será ilegal. Menos mal que no planeo viajar por allí! Esta compra no me va a salir nada rentable porque tampoco debo utilizarlo para ganar dinero. Uso no comercial, dicen.*
> *Luego en casa me he dado cuenta de que el programa usa una letra muy pequea y no veo bien las cosas. Se lo he llevado a un primo mío informático a ver si me lo podía arreglar, pero me ha dicho que no puede, que es ilegal. Ya no quiero el programa (estoy muy enfadado) pero no puedo venderlo ni regalárselo a nadie: también está prohibido.*
> *Me dice mi primo que esto se llama modelo de mercado del software privativo y que llegó a principios de los 80. Pero dice que existe una alternativa llamada software libre. Y cómo es eso?, le pregunté. Fácil, contestó, es como comprarte una bicicleta.*

7 Document Foundation

Charles H. Schulz:

> *"keeping languages and cultures in this century needs to have fully complete location software"*

Oracle bought Sun in 2009 (*lots of stories have started here since this date :)*). This affects to OpenOfficedevelopment before (Sun crisis didn't provide founds to the project), after (New company, new roles, new organization) and then, Oracle semi-abandoned OpenOffice efforts: disappeared references from web pages, guidance and roadmaps. All the work that OpenOffice was doing for 10 years was near todisappear The project will stop to exist and the project members had to prepare and founded *The Document Foundation* and the **Manifesto**

> *"For the past ten years, the OpenOffice.org community has developed, supported and promoted the world's leading open-source office productivity suite. We have attracted the support of tens of thousands of individuals and corporate bodies during this period. We now call on all our supporters to follow us into the next phase of our development, as we become an independent Foundation."*

Leaded by proposing available FLOSS projects to people. From this date The Document Foundation created LibreOffice to continue development of an FLOSS Office solution to people. This was the born of *LibreOffice*. Now available 4.0version while I'm writing these lines.

7.1 Technologies

The basic rule in a FLOSS project is *"show me the code"*. Thus as is in LibreOffice, get into the code and communication channels to create an easy development tools forge.

- *Distributed repository* - Chose git as social and cultural decision instead of other options like Mercurial or Bazaar.`http://cgit.freedesktop.org/libreoffice`
- *Integration and Revision* - Gerrit project for patches integration (not feature branch).
- *Issues Tracking System* - Bugzilla: Social and Cultural decision - `https://bugs.freedesktop.org/`
- *Wiki* - For contributors, developers, translations, desing in `https://wiki.documentfoundation.org/Development`
- *Release often Release Earlier* - Fixed release dates - 1 year and half - Have a *Public Roadmap*. Release often and you will find bugs, and this is good because maintain

developers interest.

- *License* - LGPL 3.0 and MPL allow downstream from Apache License.
- *Mailing Lists*: Developers, questions and answers, etc.
- *IRC Channel*:#libreoffice-dev channelto contact developers, users, immediatly.
- *Forums*: This channel is the most used and active by developers and users even than the mailing lists -`http://es.libreofficeforum.org/`.

There is a reference guide for first steps with LibreOffice development available at `http://www.documentfoundation.org/develop/` with all detailed references.

7.2 How to Contribute

Charles aimed us to contribute without fear to the project, trying and sending our developments (bugs, features, etc) to mailing list. He saids that all work is well received and more important the code you contribute to the project belongs to you, there are no developer agreement neither ownership transferences in LibreOffice project. *"No Pyramids, self interest"*.

To main sections where explained:

- *Easy hacks* -`https://wiki.documentfoundation.org/Development/Easy_Hacks`. *'Easy' bugs are waiting for you* in this buglist. The first door to contribute and get feedback to get in touch with this big community. These bugs are prepared for people to contribute to the project as a start up page and familiar with community process. I think is a good idea.
- *Crazy ideas* - `https://wiki.documentfoundation.org/Development/Crazy_Ideas`. The title explains a lot related to this idea, and I like it:*"With a product like LibreOffice, there is often a floodgate of radical ideas on directions the project might take"*. Its a queue to propose new ideas from brainstorming community process, they invite you to contribute in this section will your thoughts, not all will be discussed but where most ideas come together and mixed the best ideas should appear.

There is a Wiki explaining the process to get involved as LibreOffice contributor: `https://wiki.documentfoundation.org/Development#How_to_get_involved`.

As in other FLOSS projects *LibreOffice* has a basic guides to contribute in development process and decision making inside the community and how to decisions are made and which way the process become more open, simple and participatory.

- *Meritocracy* to propose and defend technological aspects but always has to be argued, always.

- *Vote for specific things* - Voting is used lesser but for some decisions is easier to vote, release dates, conferences, workshops, etc.
- *Do, don't talk* - Do-cracy, sometimes is better to do that wait or argue to introduce new features, bugs, etc. Distributed Control Systems helps to generate easily new solutions to share with community.

Process patterns:

- *Simple Tools* - As easy as only focus on your goal.
- *Simple contribution process* - You came with your patch and present it to the community. No Agreements to give code owner rights, belongs to you.
- *Loose structure (teams)* - I could say scalable teams, with a good workflow team are mutable and easy to working with.
- *Native language* - Teams duplicate whatever they feel is useful for their project. Free to replicate for every language by each team.
- *Transparency* - The most important aspect, transparency, to show the work and be truthful to everyone.

I suggest you to visit Charles website and Easy Hacks section.

8 Libre Desktop

This is a different talk from others saw in the blog. *Libre Desktops and Public Administrations* talk tries to measure FLOSS *(Free Libre Open Source Software)* Desktop environments success in Spain Public Administration.

Main goals of mixing FLOSS software in projects could be resume in:

- Balance costs
- Reinvest in *"Comunidades Autnomas"*(CCAA).
- Develop own solutions.

Encouraging FLOSS use in Public Administrations becomes a double success for projects and CCAA.

Yes because FLOSS has to be encouraged and must use in every Public project for each country, because is paid with people's money.

But is not easy as it seems, because we have the responsability to spread the benefits of using FLOSS licenses, projects, documentation and help possitively to adopt these kind of advantages.
FLOSS doesn't mean GRATIS. I recommend you take a look this disambiguation in FLOSS projects that I made before.

8.1 Transparency

Transparency is a very important word inside FLOSS world and has to be as important in Public Administrations world. All public data belongs to everyone in the country, so starting from this assertion we see the needs of FLOSS licenses in Public Administrations to continue evolving with solutions, adapting our needs and learning from others development. It's the wheel of knowledge, allways hungry.

8.2 Ministerio de Administraciones Pblicas

Miguel Angel Amutio Chief of Planning and Operations area at *Ministerio de Administraciones Pblicas*made a study to introduce FLOSS advantages inside CCAA project development in 2005. This translation from FLOSS to human-readable content is necesary to spread FLOSS advantages inside a new area.

Main topics could be summarized in:

- Overall review of what is FLOSS to define, explain and put into operation in public administration.
- Representation of Software as a Service.

- Give a chance to negotiate from FLOSS away from slavery to proprietary software.
- Investment protection against a Single vendor that can disappear.
- Interoperability.

They aren't weird and extrange for a Software Developer but, for common people (not Pulp's song, the others :))away from the FLOSS world, is not so simple to know these benefits or advantages that it provides. Yes, it provide us, I speak as a citizen which my (our) taxes are invested in IT solutionsdevelopment for public administration.

"Liability management software from FLOSS"

You can see the whole article in Administración Electrónica website.

This was a very huge step for introduction to FLOSS in Public Administrations in Spain. During those years invest in FLOSS projects grown producing, better solutions, Linux distributions, and training,devoting more resources to the knowledge than the payment of licenses, ie encouraging Public Administrations knowledge.

8.3 Cenatic & Libresoft

In 2008 Cenatic and Libresoft made an analysis to measure FLOSS implantation success in Public Adminstrations to retrieve more data directly from software developers, not only 'static' data from code.

- Quantitative analysis of the public administration.
- Based on the experience of government employees (interviews).

Here is the complete document.

After all data retrievement IMHO *(In My Humble Opinion)*, the most important thing is to spread the virtues of using FLOSS in public administrations because this advantage is like a boomerang, closes the circle of knowledge,returning knowledge to people who invest in the creation and use of a public service.

Must be able to explain the virtues that gives us the use of FLOSS in any field on the basis that knowledge is free.

9 Wikipedia

I'm not going to introduce what is Wikipedia, you know what Wikipedia is. I will write about what is not and how Wikipedia community is governed by rules using tools for this mean.

Everything revolves around *"citation needed"*quotation.
Miguel Vidal explains the Wikipedia community during the talk through the vision of a Wikipedian. A quick look at what is not Wikipedia

- Is not a dictionary.
- Is not a blog.
- Is not a centralized image bucket.
- Is not a web of links.
- Is not a collection of meaningless information.

Wikipedia is a Free Encyclopedia *(yes,I failed, I finally described the meaning of wikipedia :))*with Creative Commons By Share Alike (CC-BY-SA) License.

9.1 Technologies

Wikipedia, Wikipedia, Wikis everywhere ! What was before ? Wiki or Wikipedia ? Wiki was created in march of 1995 byWard Cunningham expert software developer, pattern designer and extreme programing pionner. He defined Wiki as *"The simplest online database that could possibly work"*.

Wiki is a content management system focused in team collaboration development. Easy collaboration and visualization for a wiki gives disseminated widely results among developers. The key point is the change history associated with each of the pages. The tool allows you to navigate through the story and find out who did what and when. The more information is saved, the more will learn. Then in 2001 he received a big boost from wikipedia to be chosen as the basis for creating the free encyclopedia that during the 2000s was on everyone's lips (anyone who knew the wikipedia), developers, architects, lawyers, children, teachers, etc..

The use of wikis is widespread in software development projects as it allows the integration of collaborative document usage. It uses a simple markup language to represent the text in an agile as a html.

The Wiki used in Wikipedia is *MediaWiki* licensed under the GNU GPLv3. It was developed specifically for the project by Magnus Manske. Today has many users who create their Wikis with MediaWiki software.

Although the only lack of wikis, is *markup languages unification* and therefore the need for *WYSIWYG* (**w**hat **y**ou **s**ee **i**s **w**hat **y**ou **g**et) editors for each type of language. At the end, allows to choose one that best suits for each needs.

9.2 How to Contribute

In Wikipedia contribute is easier that it seems. Go to a Wiki ie Software release life cycle and click *edit* button, add your content and save. That's all, no registrations, no forms, no personal data, edit and save.

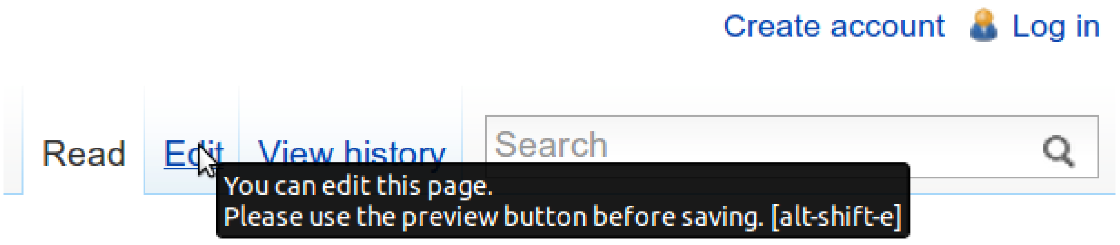


Figure 7: Wikipedia edit button

This is your first*affaire*with Wikipedia could be the last only if you want.

Wikipedia has a dedicated domain to Community in Community Portal:

> *"The Community portal is a place to find collaborations, tasks, and news about English Wikipedia."*

There is a guide to start to contribute to Wikipedia throught different ways.

Important: Wikipedia is divided in sub communities per each language, this sample is from English Wikipedia.

- *Help desk* guides users to learn how to edit using Wiki syntax. How to create an article, describe references, links, citations, relate articles, how to write a Wiki.
- In *Reference desk*; you can ask questions to librarians to guide you through Wikipedia inside articles.
- *Peer editing* help is a menthoring program for newbies in Wikipedia called Tea-House. Very interesting part in Community to guide users not only with described rules in more Wikis, community users help to others to interact and use Wikipedia hand by hand with a menthor from Wikipedia. Human communication at least. Very important and appreciated.

- *Village pump* is a discussion forum for Wikipedia technical issues, policies, proposals, new ideas. Community discuss policies for Wikipedia reasoning to increase Freedoms in Free Encyclopedia Open Community.
- *Dispute resolution*, always are disputes, complains, differences between contributors. This guide tries to resolve them before they exists explaining rules of respect inside Wikipedia and in case you get in the middle of a flame discussion how you can handle this problem and try not to become personal. Always guide yourself from your reasonings.

I hope you enjoyed and discovered new sections in Wikipedia because is not just an Encyclopedia (Free Encyclopedia :)), is a big community helping each other sharing knowledge with everyone. *People helping people.*

10 Liferay

We can read in liferay.com:

> *"Liferay Portal is an enterprise web platform for building business solutions that deliver immediate results and long-term value."*

Translated to human language Liferay is a FLOSS Portal Management System that tends to Content Management System. Nowadays CMS and PMS definitions are converging.

Liferay is oriented to Enterprise Portal Managemet. Its main costumer target are companies that purchase **Enterprise** product and **Partner** solutions.

Liferay encourages community to generate upstream and communications inside the project.

10.1 Technologies

Liferay is developed in Java and uses Eclipse Liferay IDE for developers. Also needs Liferay Portals and SDK to start developing. With this complete pack you can start developing your own portlets, themes and extensions for Liferay.

There is a guide to develop Liferay in Ubuntuexplaining from installation to deployment process.

Liferay uses Git and has GitHub profile -`https://github.com/liferay`. They use GitHub as code reviewer tool linked with JIRA issues and Forum questions.

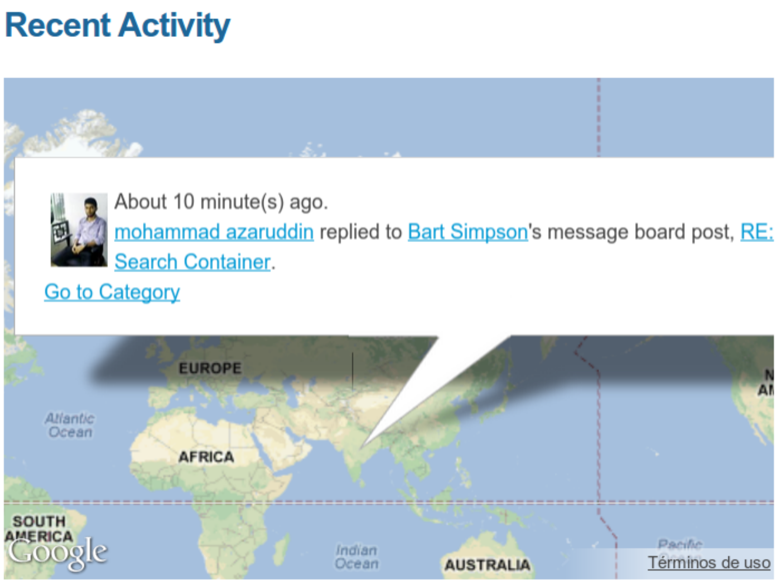
Here is the picture:

- Read Documentation guidelines.
- Install Liferay Development Environment.
- Search for an Issue.
- Develop a solution using guidelines.
- Make a *pull request* from GitHub to review code.

Technologies used in contribution process are: *Java, Eclipse, Git, GitHub, Forum and JIRA*. I encourage you to take a tour in next (work in progress) Liferay Development, *OSGI Migration Documentation.*

10.2 How to Contribute

Starting from Dashboard first thing you find is a map of global contributions:


Recent Activity

About 10 minute(s) ago.
mohammad azaruddin replied to Bart Simpson's message board post, RE: Search Container.
Go to Category

Figure 8: Liferay Contributions Dashboard

Our firs impresion is that Liferay community is developed worldwide. They publish community analysis, workshops, releases and community success samples.

It's important to start readingLiferay user guide, everyone should start from here, because you need to know first what Liferay is.

Liferay encourages users to create a profile page, as we can se Jorge Ferrer Liferay personal page. Gives importance to create a 'centralized' community.

Like other FLOSS projects they divide community section in two main groups:

- Participate -http://www.liferay.com/community/welcome/participate
- Contribute -http://www.liferay.com/community/welcome/contribute

Participate section invites the user to share the experience of liferay, doubts, achievements, proposals, errors found, so that people who encounter the same mistakes do not recur. Thus, creating a place to clarify questions and share knowledge, Liferay community.

You can find different ways to communicate with Liferay community, *communicate is as valuable as contribute with code*:

- Community Forum
- Blogs
- Wiki

- IRC
- Liferay LIVE
- Events
- Issues
- User Groups

In *Contribute* section you have more technical information about how to contribute 'explicit' with community. Wiki section explains different ways and each guide defined to contribute. There is a wide fork to contribute, from translations to writing a complete Liferay plugin:

- *Reporting and/or fixing a bug*: JIRA ITS is used to provide easy bug/issueTS interface for users. Explained detailed how to report a bug, publish it, search for it and develop a solution.
- *Writing documentation*: In Wiki section is described that every user can write Liferay documentaiton. There are two main guides *Wiki Guidelines* and*Liferay Editorial Guidelines*to contribute with documentation.
- *Implementing a new feature*: Section Proposed Projects to describe your solution after read and develop your new feature described in trunk Development Guidelines. Also you can suggest a new feature in Liferay Forum.
- *Providing Translations*: Liferay Translation Teamhas its own guidelines to manage translation teams for each Language.
- *Writing a useful plugin*: They provide a complete developers guide for Liferay plugins in pdf. You can download it here.

Ohter remarcable section is Community Feature Ideas. The user can provide ideas to Liferay evolution. Manages JIRA to advertise for Liferay different ideas,JIRA becomes more social and guided by the votes of the people to develop new ideas.

11 FLOSS Migration

How is possible a FLOSS migration in Public Administrations ? Here we have *Zaragoza succes case.*

ZO2-AZLINUX:FLOSS migration project designed and developed from 2005 in Zaragoza. Inside description we found their slogan:

> *"Migration from proprietary software to free software on the computers of municipal employees."*

Migrate private software to FLOSS in Zaragoza Public Administration gradually emphasizing the migration office tools *Microsoft Office* for *OpenOffice* and *Microsoft Windows* operating system for.

They developed a migration plan including all actors: politics, technical and users. The whole path to achieve this goal, that all administration could work using FLOSS solutions. Thus Public Administration developments and maintainability is not locked to any private software provided and could invest in Research and Development.

This plan is included within a larger plan. Which Zaragoza tries to turn one of the major European cities with greater economic development based on new information technologies and knowledge management.

Zaragoza city of knowledge:

`http://www.zaragoza.es/ciudad/conocimiento/conocimiento.htm`

11.1 Technologies

Technologies involved to provide same functionality that private software solutions:

- 2005 - 2007: Browser Firefox, Thunderbird, Multimedia. Changes soft for positive acceptance.
- 2008 - 2010: Office Suites, OpenOffice. Change is harder but accepted.
- 2009 - ˜SO: Switching to OS AZLinux.

These steps **must be accompanied** by a parallel track:

- *Inventory*: Hardware and Software.
- *Communication*: Managing change. Accompany the users through the traumatic process. Talks, cd testing, information exchange process, benefits of using FLOSS
- *Technical Training*: An important point to consider.

- *Users Training*: 8 hours Windows xp to Linux - 20 hours Office to OpenOffice. *Remarkable time dedicated to users training related to Office Suites higher than OS change.*

11.2 How to Contribute

How could you contribute to this process ? I think the best way to start contributing is to spread this work and its success case to people reluctant to switch to FLOSS. This way is that we have to follow, thus the main reason is the freedom to choose that FLOSS gives you as a user and of course the ability to spread and share knowledge to everyone.

After those principles, I want to highlight these links:

- Activities -http://www.zaragoza.es/ciudad/sectores/tecnologia/swlibre/actividades_swlibre.htm
- Open data initiative -http://www.zaragoza.es/ciudad/risp/
- Promote your applications with open-data -https://www.zaragoza.es/ciudad/enlinea/consulta_risp.xh

"Open data and FLOSS to serve the people"

12 GNOME

GNOME is an acronym that means: *GNU Object Model Environment.* Was created in 1997 through the visible momentum of *Miguel de Icaza* to provide graphical desktop environments completely FLOSS Linux version build using components and focused on interoperability. Counteract the existing KDE fed on QT (at that time **was proprietary software, not now**). After some analysis they start developing GNOME using GTK (*Gimp Toolkit, yes Gimp GNU Image Manipulation Program*:)).

GNOME Foundation was founded on 15 August 2000 by Compaq, Eazel, Helix Code, IBM, Red Hat and Sun Microsystems. Manages GNOME development and represents GNOME worldwide.

12.1 Technologies

GNOME is developed using C language and Object Oriented GTK+ library. Each project/module is governed by these basic development tools:

- *Maintainer* - benevolent dictator for life for each module. GNOME has a distributed development management divided in modules.
- *Mailing list* - Encourage users to develop discussions in mailing list instead of IRC because it is easier to manage the history of the decisions made. Mailing Lists.
- *Bug Tracking System* - GNOME Bugzilla manages Project development, Bugs and discussions.
- *Documentation* - Wiki is a very useful tool in project development because brings information from the project and its easier to use as a collaborative document development for technical and non-technical users. GNOME Wiki is published in `https://live.gnome.org/`.
- *Repository* - Distributed repository git in `https://git.gnome.org/browse/` and a user guide that explains the use of the repository in live Wiki.
- *Synchronous communication* - IRC channel to retrieve information and ask questions related to GNOME. You can find more information in Wiki IRC section.
- *Quality and Assurance* - Users, the users are the best QA team for Gnome.

There is a manual for developers specifically dedicated to them, GNOME Developer Center. In showing a series of tutorials, documentation and GNOME structure to facilitate the use and implementation of the platform.

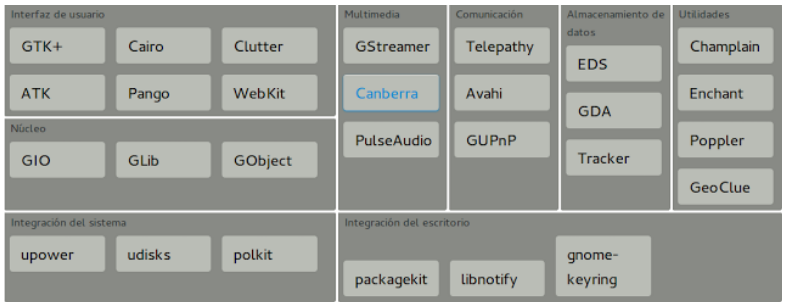


Figure 9: Gnome platform overview

12.2 How to Contribute

GNOME Community has a low barrier entrance guided by Meritocracy. Community guides are available to everyone in GnomeLove. Here we are going to describe internal process of a contribution and how a contributor reaches the goal of becoming a commiter througth different organization ROLES:

- Reasonable number of patches in a module.
- Ask for commit rights to Accounts Team
- Accounts Team contact module maintainer or translator coordinator to check for approval.
- A committer can commit to git repository but her contributions have to be reviewed and approved by maintainer of the module
- **Important**: committers have write access to all git repositories hosted in git.gnome.org. You can contribute to all modules. If you find an error you could fix and commit the patch.

Maintainers are responsible for the module. They have to *review patches* and *assume results*, their role is important inside the community have high responsability. Communications and module releases management are managed by maintainers. Could be various maintainers for one module.

Instead of development committers, translators committers in translation modules also are maintainers for themselves because *a translation can't be reviewed* because *it would duplicate the work*. So you have full confidence in the translation module.

13 Thunderbird Q&A

Thunderbird is a FLOSS Email client from Mozilla Foundation - *MOFO* - . Latest news aren't good news because Mozilla stop development resources to this project on July 2012[1] but this talk showed us the way developers work in Thunderbird and how Quality & Assurance - *Q&A* - Team and developers join efforts to lead a better development.

Ludovic Hirliman[2] is*Q&A*leader in Thunderbird, through his vision we are capable of see the big picture behind good software development starting with Test Driven Development - *TDD* - .

"Testing Tools: your hand, your brain and time" by Ludovic Hirliman.

13.1 Development Tools

"Someone wants to test ? go to url, test and report bugs"Focusing in our goals, here are Thunderbird development tools:

- *Bug Tracking Sustem* - Bugzilla - Everyone could post a bug and how to replicate a bug there.
- *Test case management software* - From Litmus to Moztrap. Tools to create and run test cases related to email clients.
  - Limus - Test interactively with real email clients.
  - Moztrap - Test Case Management from Mozilla.
- *Continuous Integration* - Buildbot - Checkout, run make, run make test, read log file and automate a build in every commit. Running unit test to build after each commit.
- *Documentation* - Wiki to develop extensions and a real user Wiki.
- *Communication* - IRC as main communication channel, group chats, history in#tb-support-crew.
- *Testing Tool* - Selenium - Framework to automate integration browser testing.

After throw away all spaghetti code in 2009, next QA goal in January 2010 was: "To add code to Thunderbird developers have to develop tests". After those steps the project obtain more volunteers. The code you write matters and its easy to contribute, collaborate and test.

[1]Thunderbird finale - `https://blog.lizardwrangler.com/2012/07/06/thunderbird-stability-and-community-innovation/`
[2]Ludovic Hirliman web - `http://perso.hirlimann.net/~ludo/`

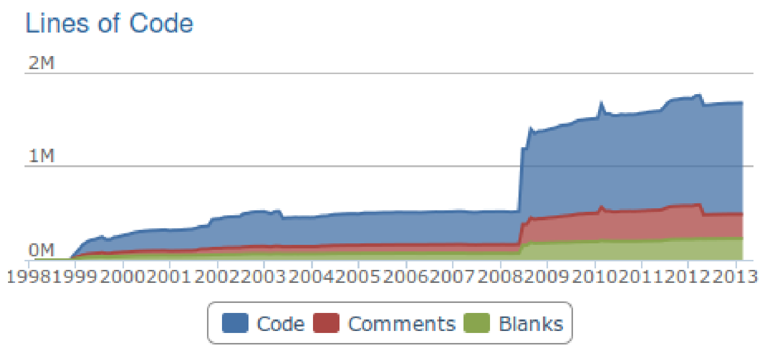


Figure 10: Thunderbird code evolution

Thunderbird lines of code evolution jumps in 2009, filled with test classes

13.2 How to Contribute

There are many ways to contribute to Thunderbird, be a user is the first step to dive into the community:

- *Development* - Reporting bugs, patches and contributing with code.
- *Q&A* - After failling MozillaFriday test dayprocess into Thunderbird community, they changed the question to: *"Do you want to contribute with test ?"* Thus they manage people and explain what and how they 'have to' test. Its an easier way to start to collaborate instead of *"Someone wants to test ? go to url, test and report bugs"*. Better and easy as human way.
- *Support* - Translations and localizations.
- *Marketing* - Mozilla products Evangelist. Spreading the goods and benefits of MOFO products.
- *Superheroes* - Thunderbird Contributors starting pageinmozillamessaging.
- *Forum* - Thunderbird forum to search and generate information.

14 Canonical

Canonical is the company that launched Ubuntu project in 2004. Founded by South African Mark Shuttleworth after a profit of 500$ million from the sale of his security company. Decided to invest the money to create a Linux distribution more friendly and easy to install

"Linux for human beings"

So, he got in touch with Debian developers (which is one of the most reliable Linux distributions) to offer financial support to create a more human Linux *Ubuntu*.

After a controversial year because of this distribution is owned by one person, decided to create a Foundation in 2005: Ubuntu Foundation to prevent Ubuntu depended on a person or organization and was now to be directed by people. That gave decision making to community and Copyright Holder of Ubuntu to the Foundation to avoid possible distribution privatizations by a Company or a person.

14.1 Technologies

Canonical is not a software solution is a company that develops FLOSS project Ubuntu and other solutions that are not FLOSS.

We can take a look to Technologies used to develop Ubuntu with the community. Explained in their Wiki we summarize main goals in Get involved section and focusing our summary in Kernel development Team:

- *Communications*
  - *Blogs* - Here is a planet that unifies all development team personal /technical blogs in one feed releated to huma project members in http://voices.canonical.com/kernelteam/.
  - *IRC* channel #ubuntu-kernel to get in touch with users and developers. Weekly IRC public meeting in #ubuntu-meeting to strengthen collaboration within the project.
  - *Mailing List* - Mailing lists are highly used in Ubuntu, you can subscribe to contribute here.
  - We run our mailing list much like the Linux Kernel Mailing List (LKML). Users can propose patches, ask questions and find general info. This list has medium traffic volume. For more info you can read https://wiki.ubuntu.com/KernelTeam/Kernel
- *Launchpad Teams* - ALM Tools from Ubuntu to deploy projects, I/BTS (Issue/Bugs Tracking System) and public roadmap.
  - Ubuntu Kernel Team this is the moderated public team in which we interact with the Ubuntu Community.

  - Canonical Kernel Team this is a private team used by the Canonical Kernel Team for Canonical paid hardware enablement and other Canonical business.

- *Contacts* - This section is special because translates developers to humans to get in touch with them, is based on the philosophy.

14.2 How to Contribute

Ubuntu Development describes a guide of how to contribute, collaborate and communicate in team projects. Ubuntu describes a community Code of Conduct. A must read document that explains relations inside Ubuntu community, respect, responsible, collaborate, teamwork, meritocracy, ask for help. All rules to guide a respectful community teamwork based in communication and consensus. You **have to sign this contract** to contribute to Ubuntu community.

Encourages you to subscribe to mailing lists and start reading information about the community and how work internally.

After that take a look to bug section managed by BugSquads. Here you can contribute searching and reporting bugs by your own or join a bug team specialized in a module. Bug Squad has a guide of their job described to triaging:

- Responding to new bugs as they are filed.
- Ensuring that new bugs have all the necessary information.
- Assigning bugs to the proper package.
- Confirming bug reports by trying to reproduce them.
- Setting the importance of bugs reports. (Bug Control members Only)
- Searching for and marking duplicates in the bug tracking system.
- Sending bugs to their upstream authors, when applicable.
- Cross-referencing bugs from other distributions.
- Expiring old bugs.

This section is very interesting because they are like bug soldiers inside a QA *users* team and have a clear and strict workflow to deal with bugs.

Ubuntu Developer Summit is an open-innovation section:

> *"The bulk of UDS is discussion sessions. We explore problems and develop solutions together, pooling our collective experience"*

Includes developers, proposals, technical users through the common idea to unify roads and problems, to create a team, nearly through the exchange of ideas and views. A good proposal for collaboration in the community.

15 OSOR

Open Source Observatory Repository - Free Libre Open Source Software - *FLOSS* - Forge oriented to European Public Administration. Giving a technic base and guidelines to develop FLOSS projects throught good practices. Projects has to be licensed with EUPL European Union Public License. *Only* Public Administrations funded FLOSS project can be hosted in OSOR you can not advertise a project by yourself.

The project started in 2007 and in December 2011 migrated to JoinUP the new development platform.

15.1 Development Tools

OSOR main goal is to create Communities for each FLOSS project thus create an environment in a common place to all Public Administration developments.

- *Gforge* - Base for the FLOSS Forge that became Fusion Forge in 2009 (http://blog.bitergia.com/2012/1 history-of-fusionforge-and-gforge/). Strong communication with Gforge community.
- *Repository* - Apache Subversion for source code management.
- *CMS* - Service for users Plone CMS
- *Mailing list* - GNU Mailman.

I couldn't found more information about development tools for the project.

15.2 How to contribute

The project ended in December 2011 so is not possible to collaborate in this project, there is no community supporting the project. Is a FLOSS project thus everyone could continue its development or play with the code to learn.

This summarizes community management ROLES big picture inside OSOR project A.K.A. *Cheerleaders*:

- **Cheerleadears** - People dedicated to spread OSOR and advertise the goodnesses of using OSOR.
  - *News* - Gijs Hillenius - Journalist specialized in FLOSS and communication with other FLOSS communities, promoting workshops, events and meetings between communities.
  - *Coordination* - Unisys Ismael Olea (@olea) spreading OSOR use in Europe.
  - *Dynamizer* - GOPA

16 KDE

KDE is a Desktop ? an environment ? What is KDE ? KDE started being a Desktop environment. Was founded in 1996 by Matthias Ettrich looking for the creation of a working and intuitive Desktop Environment for Unix distributions with common applications to create an ecosystem.

The projects evolved using Qt framework which became FLOSS software in 1998 and the first version KDE 1.0.

Continued evolving and growing with user until this transformation in 2009 into a FLOSS developers community, not only Desktop environment. Thus KDE wanted to spread the use of Free Software in desktops, without focusing on the development of the desktop environment. The foundation covers most fields of product development for desktop. Thus trying to spread the use of Free Software in all levels of users. It is an ambitious but successful.

16.1 Technologies

Development in KDE is divided in different sections:

- *Core Tools* - CMake and VCS tool, Git or SVN. Basic tools to build.
- *Debugging and Analysis* - Tools for debugging KDE projects are composed by Valgrind, The GNU Project Debugger (GDB), KDbg and DDD, MS Windows tools.
- *Development Tools* - A set of IDEs to develop KDE projects: Qt Creator, KDevelop, MS Visual Studio Express IDE (Windows only)
- *Internationalization (i18n) Tools* - Set of internationalization tools to contribute easily - Lokalize, Dr. Klash, The x-test language.
  - *Examining .po files*
- *Helper Tools* - Get information about KDE's installation - kde-config, Driving Konqueror From Scripts - kfmclient, Updating User Configuration Files - kconf_update, Generating apidox, Automoc4, svnmerge.py
- *Quality Assurance* - Code Review, Continuous Building, English Breakfast Network - Static Analysis

Detailed information can be found in http://techbase.kde.org/Development/Tools. TechBase KDE is a place made to share knowledgement for everyone. Was created after KDE became more than a Desktop Client (first edition of Development Tools in 2006).

16.2 How to contribute

How to become a committer ? You can ask any community member for this permission. It's a community with easy entry and likes to receive new members.

After this point, you have to work on it following community guidelines. The steps aren't different from other communities, we can see a common pattern in communities:

- *News and Mail Sources* - Mailing lists, history and news.
- *Reporting Bugs* - Bug reporting and bug fixing.
- *Getting Started with Coding*: C++, Qt, KDE - Yes, you have to code to contribute to development.
- *Getting Involved in Bug Hunting and Application Quality* - Be a Quality Assurance expert helping improving test and quality.
- *Junior Bugs* - This is a special part to get in touch with the community, an easy way to start fixing bugs and know how to work in community.
- *User Interface* - Avoid flames about UI *"User interface is a very wide subject".*
- *Getting Answers to Your Questions* - Learn to search and use documentation. It's very important to read the FAQs, mail lists, documentation, wikis, bugs before post anything because could be duplicated and become a nonsense effort

Extended guide could be found at http://techbase.kde.org/Contribute.

17 Differences and Similarities

Using matrix visualization we are going to appreciate differences and similarities between contributions access guidelines through project communities analyzed in talks checking:

- *Mailing lists* - Mailing list such as developers, users, etc…
- *Developer Guide* - Documentation for start development environment.
- *Bug Tracking System* - Bug or Issue tracking system to manage project resources and schedules.
- *Mentoring program* - Mentoring program to introduce new users to produce upstream in the community and guide them into the process with another members of the community.
- *Forum* - Forum, FAQs or other tool to interact the community creating community.
- *Documentation* - Project documentation for every type of user, from technical to end-user (if exists).

Y - *Yes*, **N** - *No*, **N/A** - *Not Available*

	Mail list	**Dev Guide**	**BTS**	**Mentoring**	**Forum**	**Documentation**
Apache	Y	Y	Y	Y	**N**	Y
WebKit	Y	Y	Y	**N**	**N**	Y
Plan9	Y	Y	**N**	**N**	**N**	Y
Mozilla	Y	**N**	Y	**N**	Y	Y
GNU	Y	Y	Y	**N**	**N**	Y
Open Document	Y	Y	Y	Y	Y	Y
Libre Desktop	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**
Wikipedia	Y	Y	Y	Y	Y	Y
Liferay	Y	Y	Y	**N**	Y	Y
AZLinux Migration	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**
GNOME	Y	Y	Y	**N**	**N**	Y
Thunderbird	Y	Y	Y	**N**	Y	Y
Canonical/Ubuntu	Y	Y	Y	**N**	Y	Y
OSOR	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**
KDE	Y	Y	Y	Y	Y	Y

18 Conclusions

Taking a first look into the result table we can remove data that can't be evaluated from projects such as:

- *Libre Desktop* - The talk was related to how the law evolved to IT and FLOSS during last 30 years in Spain.
- *AZLinux Migration* - Migration to FLOSS in Public Administrations. This project as is explained is more important to spread this results that contribute because is a process not a FLOSS project by itself.
- *OSOR* - OSOR Project disappear in December 2011 so, development and contributions doesn't already exist for our misfortune.

The following leads out to *Mentoring* column. There are a lot of projects (analyzed in the talks) that do not involve mentoring 5 to 7. Those projects that use some kind of mentoring (interactive manuals guided by a another user/s) are more focused to end-user (non technical) that others. Their marketing is related to more 'tangible' projects for the 'real world' like Wikipedia, KDE, Open Document and Apache ? I think they have more 'human' part than the others and because of that they produce these mentoring programs.

Other remarkable point are the *Forums*. Surprised when I saw the results that projects like GNOME doesn't have official forums, neither than WebKit a project used worldwide and covers many fields. These projects work with what they need, it is always better **to know that you do not need**.

Jono Bacon in Art-of-community introduction to *Communicating Clearly* chapter:

> *"Community is absolutely about understanding the ether. Our notes are the processes, governance, tools, and methods in which we work together. The notes we dont play are the subtle nuances in how we pull these notes together and share them with one another. The space between the notes is communication."*

In the opposite hand we have Open Document community that works great with them forums as Charles told us.

In closing, I wish to emphasize a conclusion on the final paragraph. The communities studied have their little rules to run through his life. As people also find it difficult to change because all change is associated with ignorance and uncertainty. What I mean is that each one works in a way through standards but underneath all, absolutely all, are communities of people who rely on the cooperation and communication between them.

We can not draw generic standards for the communities studied with the data we have exposed. It's a small study on a roughly speeches in the course classes 'Case Studies II' and the amount of research work on FLOSS communities.