
COMM.
POWER
Contribute to Open Source
THE RIGHT WAY
GPLv3 licensed - 2022.1
BY
DANIELE SCASCIAFRATTE
B
A
SELECT
START

Contribute to opensource: the right way 3nd edition

Open Source is a trending topic today but how you can learned all or some of the skills required to create or contribute to a project opened to the world? This book is my experience in the last 12 years, what I learn and what are the best practice from someone in the field.

Daniele Scasciafratte

This book is for sale at http://leanpub.com/contributetoopensource-therightway

This version was published on 2022-09-02

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

How to read this book

I have always wanted to write that sentence and this book allowed me to do it!
This is not my first e-book. Over the years, I have written a few niche books in Italian. They are probably unknown to many, but writing them was a fun learning experience for me.
During high school (2005-2009), I wrote "Principi di programmazione" (Development Principles) and I released the second edition years later, I wrote it mainly for myself, but opted for a public format for it.
Later, I wrote a first and second edition of a 12 page guide on Python and Qt[1] in Italian.
I have written also a guide of few pages "WordPress per il cliente"[2] (WordPress for the customer) in 2012.
This is not my first rodeo. I have written technical books before. This time, though, the topic of this book is broad and relevant with a lot of my opinions and the explanation of my personal approach to things.

What you will find

The book follows my mind flow from why and how I started and what I learned, based on my lectures (books or online articles) but also a lot of experience, in few words this is my view about the topic.
In this way you have a (big) tutorial in the open source life to individually create your path.
There will be memes (I love them) and funny images like my writing style on the blog or talks, because this book is written in an informal way.
If you want to read it because *I need to learn, make money and became famous* here you will find nothing for you, check on Udemy or somewhere else.
A person not engaged, not interested on the long term is already failing on contributing.

One real example from the Linux Kernel, a message by Qu Wenruo, with relative HackerNews discussion[3] shows us sometimes contributions are just a number. This number doesn't mean quality or can be a time-consuming task for maintainer (we will see in the book who they are). A company see a single change as a number but can be just a typo, that is good especially for newcomers as they are learning with something simple, but if you are a skilled person and you are doing it a lot with

[1] https://daniele.tech/2013/04/ebook-introduzione-a-pyqt-1-edizione/
[2] https://daniele.tech/2014/10/wordpress-per-cliente-rev-0-0-4/
[3] https://news.ycombinator.com/item?id=27629366

a single request for any change instead of a single one (so the changes amount that you did for the company grows) you are wasting the time for the maintainers to focus on more important things that can require their best attention.

The value is not in the contribution per se but for the contribution content, if you want really success in Open Source, and not just be a number a bit of ethic is important, as everything is public and not hidden behind a NDA/CLA (sometimes).

I like to generalize often in my thoughts and discussions. So this book can seems like that but is my way of thinking to simplify the issue or the discussion itself. As a developer is important to see the real problem and the frame but they don't need to be managed together.

What are the hints of this book

Information, suggestions, procedure, workflow, best practices.
It begins from the basic stuff to my (new) way of doing things and intro to do the first steps.
The technical terms will be explained (the first time) to help Newbies and non-tech people at their first approach to the matter.
The open world is *open* to everybody so this book will serve to explaining things.
It includes also a lot of links to external resources so you can give sources but also learn more about specific topic without going off topic with the book.
Also it is a way to give a bit of critical thinking about stuff and other point of view around open source, it is not so simple as you think!
See you at the index!

License

This book is licensed under the GPLv3 license because I like it, maybe not the best one for a book but is like a mention to remember how much this license changed the world.
Includes also memes (usually in bad quality) that are used freely following the Article 13 of the EU Copyright Directive[4] that excludes them for the copyright.
Just to mention it, I made a tool called GH-License[5] that help you to check if your repository include a license, if your repo on GitHub/Bitbucket/etc include it and to download automatically.

Remember **a project that you can download with all the code doesn't mean that is open source** if doesn't include a license. Without a license is a proprietary project!

[4] https://eur-lex.europa.eu/eli/dir/2019/790/oj
[5] https://github.com/Mte90/GH-License

Cover

Original GIMP project file (.xcf) available on GitHub[6] and published to leanpub.com[7].

- Gameboy Color image by Blueamnesiac - https://www.deviantart.com/blueamnesiac/art/Nintendo-Game-Boy-Color-Teal-438531988[8]
- Fipps Font by pheist - https://www.dafont.com/fipps.font[9]
- 8-bit pusab by Seba Perez - https://www.dafont.com/8-bit-pusab.font[10]
- 8 Bit Wonder by Joiro Hatgaya - https://www.dafont.com/8bit-wonder.font[11]

Tools used

This book is written in Markdown using ReText[12], released on GitHub[13] and published to leanpub.com[14].
The language was reviewed using LanguageTool.org[15], Grammark[16] and Sed[17].

An example of what means contribute to open source is part of this tools. When I written the first edition I did it with just ReText that has a big problem, also it has a tab UI when you open a new file from the file manager start a new instance, so I did a patch that now is part of ReText[18] that enable to use an unique instance. Working on this 3rd edition (after 3 years) with this improvement on was a pleasure compared to the first edition.

Legend

With the *open source* term usually we refer to a lot of things, from software projects to others like open knowledge (as Wikipedia), so the term (*open* or *open source*) in this book as to be intended as a big view about the philosophy of the project itself and the approach of the community.

Contacts

I am looking for **feedback** about this book to improve it, so reach me to share your thoughts and ideas.

[6] https://github.com/Mte90/Contribute-to-opensource-the-right-way
[7] https://leanpub.com/contributetoopensource-therightway/
[8] https://www.deviantart.com/blueamnesiac/art/Nintendo-Game-Boy-Color-Teal-438531988
[9] https://www.dafont.com/fipps.font
[10] https://www.dafont.com/8-bit-pusab.font
[11] https://www.dafont.com/8bit-wonder.font
[12] https://github.com/retext-project/retext
[13] https://github.com/Mte90/Contribute-to-opensource-the-right-way
[14] https://leanpub.com/contributetoopensource-therightway/
[15] https://languagetool.org/
[16] https://grammark.org/dist/
[17] https://www.gnu.org/software/sed/manual/html_node/index.html
[18] https://github.com/retext-project/retext/pull/476

- https://daniele.tech[19]
- https://twitter.com/Mte90Net[20]
- https://www.linkedin.com/in/danielescasciafratte/[21]
- https://www.reddit.com/user/Mte90/[22]
- https://www.instagram.com/mte90/[23]
- https://t.me/mte90[24]
- https://mastodon.uno/@mte90[25]

Donation

If you want to donate to the author (me) you can do it with Patreon[26], Paypal[27] or Liberapay[28].

[19] https://daniele.tech
[20] https://twitter.com/Mte90Net
[21] https://www.linkedin.com/in/danielescasciafratte/
[22] https://www.reddit.com/user/Mte90/
[23] https://www.instagram.com/mte90/
[24] https://t.me/mte90
[25] https://mastodon.uno/@mte90
[26] https://www.patreon.com/Mte90
[27] https://www.paypal.me/mte90
[28] https://liberapay.com/Mte90/donate

My (open source) story

I want to start with a chapter about me to let you understand the rest of the book (or the book itself).
I also hope to be inspirational and show new ways to approach the technology and open source philosophy.

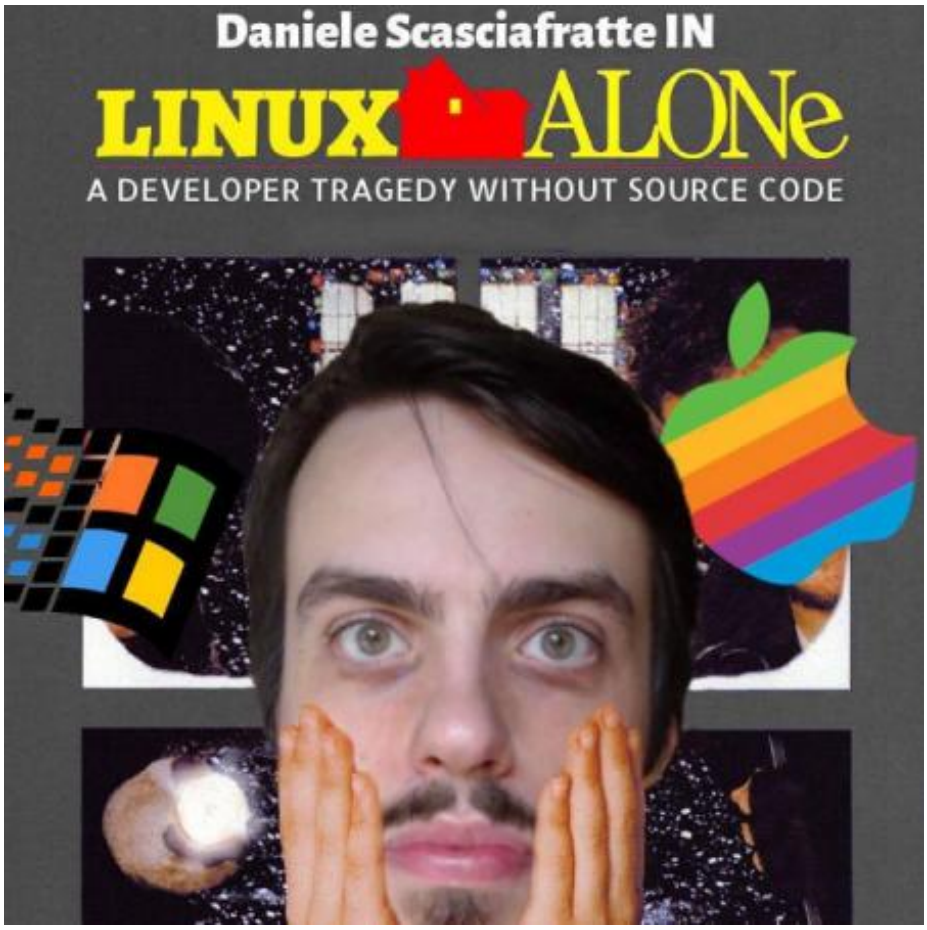
Let's start with the flashback!

I was born in 1990 in Rieti (Latium, Italy), a little city famous because is the `Umbilicus Italiae` (this is Latin and means the belly button of Italy), basically the center of Italy.
The city is near Rome, Terni and Aquila and apart from the name is famous all over the world for The Rape of the Sabine Women[29]. The legend is that when Rome was founded there wasn't sufficient women for the citizen so they kidnapped there/around. This story was famous because many artists during the Renaissance used it as theme for their arts.

I think we are more famous for the Porchetta (food) and Lucio Battisti (singer who also made stuff in Spanish, English, French and German), and San Francis that moved in Rieti and built various sanctuaries and the first nativity in Greccio.

On 1990, the year of "Home Alone", my grandmother said that as kid I was similar to that child, anyway I think I have grown a little better without thiefs trying to break in my home.

[29] https://en.wikipedia.org/wiki/The_Rape_of_the_Sabine_Women



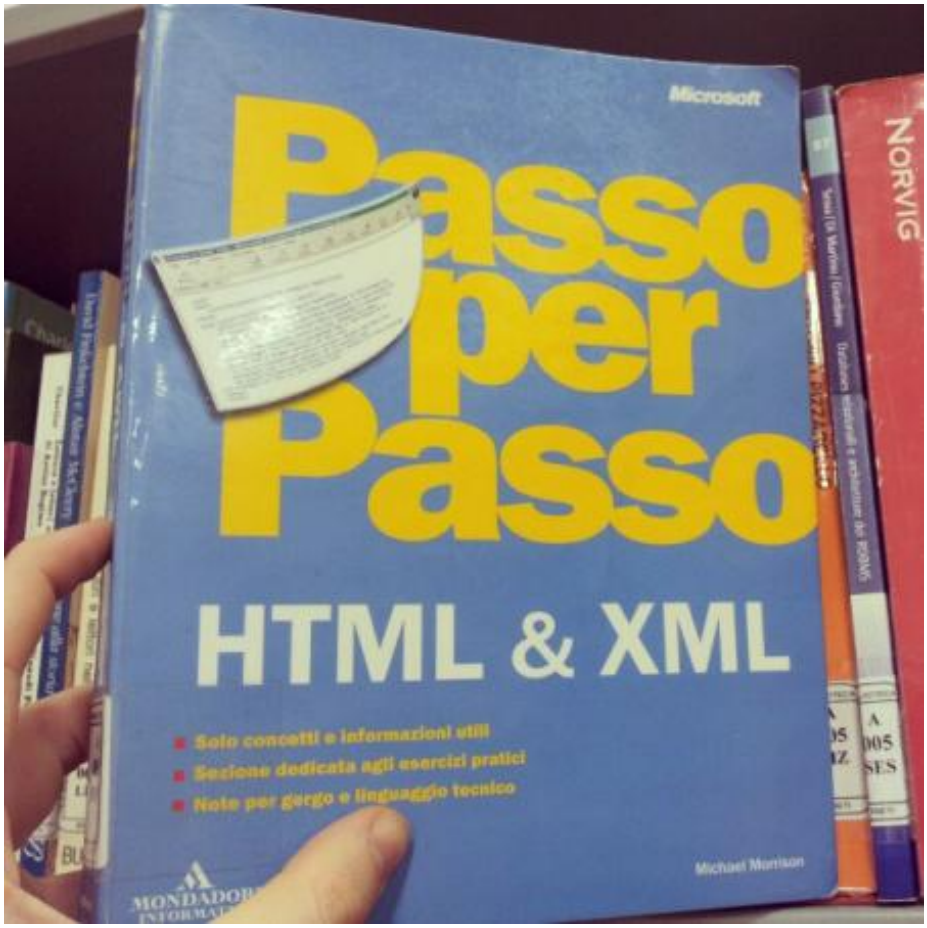
A footage that I did for my 25 years

I really like the Open Source world and this has created many problems because I joined many communities, I contributed to different projects, joined many events and sometimes this creates confusion about my role.
In my personal life I like to collect comics Italian, European and American, I have a large library with many comics in my room.
My passion about Open Source (or activism) is not clear to my friends because it doesn't revolve around sport (especially soccer that I hate a lot) and in Italy it seems that if you aren't interested in following sports (or practicing them) you are like a loser. Anyway this create issues with my friends that doesn't know what I do for living or why I travel a lot.

This is probably the first reason why I am so easy to get involved, because what I do in Open Source has a real effect on the world and people, compared to this kind of interest that for me is not productive.

I attended the "Liceo Scientifico Tecnologico" (in Italy the school differs from the others, we have different schools based on the topics), completed in 2009 at "ITI Celestino Rosatelli" in Rieti with 68/100 as a final score of 5 years (thanks to the math that lowered my score), I did a little thesis (it's quite common in Italy for the final exam in high school) and I was only one in a class of 27

people, I even took the laptop (purchased in 2006) for a demo on Lissajouss graphs on Windows XP virtualized on Sidux (the Linux distribution based on Debian Sid, that is now dead).
I started to use Linux on my personal laptop just before the end of the school but I was distributing my dev projects online (with sources already).
I started to develop in 2006, just before starting the third year of high school, where we finally should start to develop with Visual Basic 6. The school switched to more new technologies like Windows XP (we were using Windows 98) and .NET framework (was using that already). I was very thrilled from starting to develop that in the year 2006, I was already doing my first internet websites with HTML4. At the time my data source was the local public library and the IT magazines.



My first book about web development

My very first website was about the Dragon Ball anime that I maintained until 2009 and this allowed me to learn and discover the web development after the desktop development.
Meanwhile if I wanted to use internet I had to be good at school (you know get good votes). In the house we had ADSL that was hour based for the price, so I usually got 1 or 2 hours at week. In other words I had a paper with me every day with the list of stuff to check or download that I filled every week to use internet.
Something incredible compared with today!

I remember when my father got frightened because I was taking part online in forums or by chat (MSN or IRC) with people that I had never met. I started in this period to use my nickname `Mte90`, created in 2002/2003 and still using it. That is why people meeting me at events think my name is Matteo.

Anyway I was developing my software during the school times, based on my needs, this is the first reason I think everyone needs to start from a personal need, to get motivation for everything.
I am still using this approach when I need to learn something new, I don't study because someone told me, but because I need it. A different approach to school, work and life that helps you optimize but also stay motivated, because after all there is a real reason.
So to explain what I am doing or what I am learning to others, because what's useful for me may be also for others. That even before discovering the open source world.

2014 in Pordenone

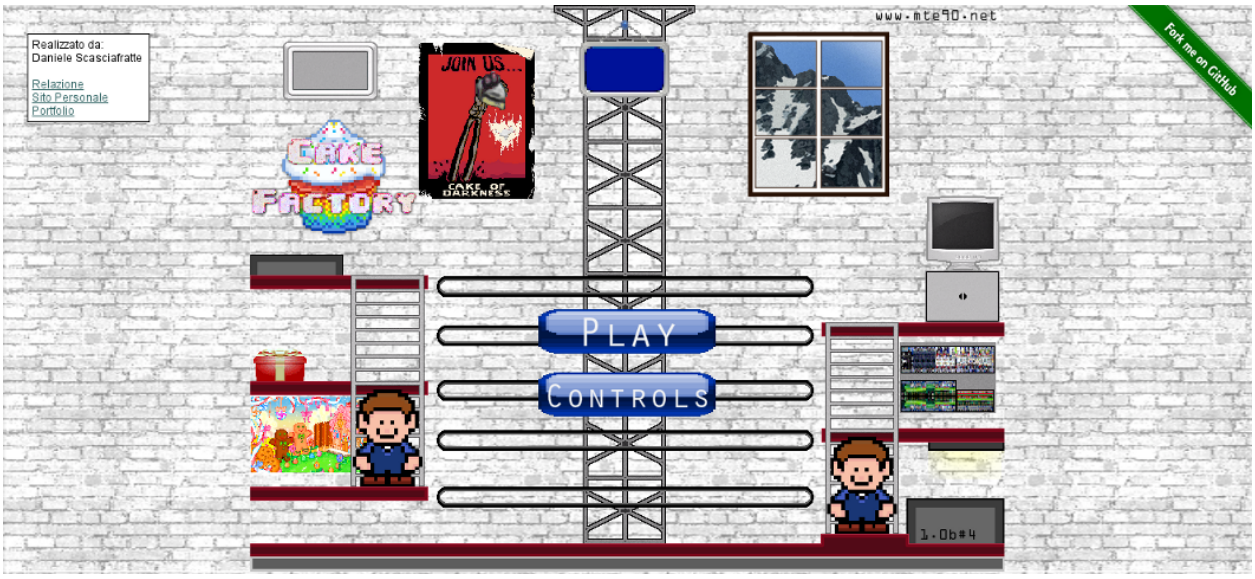
For me internet was the first way to meet people like me (I still love forums) to work on my projects or discuss.
In 2009 after the school I had to choose what to do with my life. A question that all the young people often face, my choice was very simple: work with computers (without math).
Something clearly impossible at that time, it was a like an elitarian job, because there was this

idea that to be good at computer you needed to be good at math i.e. on remembering formulas or calculating algorithms.

I chose to go to L'Aquila and take a university course at the Academy of Fine Arts on Web Design. After all I liked to work with websites, so I was easily getting the piece of paper (my parents were longing for).
For non-italians, 2009 was the year of the big earthquake with 309 dead and that course never started, so I took a sabbatical year working at the family's shop in the morning and doing a work stage in an IT company in the afternoon for 6 months.
My tasks were to install antennas for internet, configure web servers, install or fix computers; while at the shop I was learning how to talk with customers and how does a company works, at only 19 years old.

2010 was the last year I developed on Windows when I did that work phase, where I valued Linux a lot more.

Finished this year (later I worked only at the family's shop) in the position to gain the this piece of paper, that as of today is useless. I removed the reference on Linkedin and I don't want to mention the school or the course I attended for years about web/digital stuff in Rome (basically only Adobe-boring stuff).



My first real video game... in Flash

So in the 2012, when people started to call me "Linux" I started working as a freelance. Up to 2015 where I founded a Web Agency in Rome with two friends and as of today we are still doing open source development and web sites/application development.
In our company statute is written that we have to use 50% of our time on contributing to open source technologies (a big idea of Eugenio).

After 4 years we are still doing stuff but we are only two people now. With a lot of other topics to discuss in next chapters, I hope I am not being boring (hate that) and to be able to show you the key points informing my interest in this industry.

How I started to contribute to open source projects

I was developing in PHP because when I moved to Linux this was my home, with the discovery of Python and I made my first UI projects with the Qt framework. I was experimenting with doing the same things on Linux as I was doing with Windows, even with custom software.
It's basically what people do when migrating from the first to the latter, in my case I felt like Indiana Jones, without boundaries, and it was intriguing.

At 18 years (2008) I bought my domain `mte90.net`, after some time I opted for using a CMS, and since databases were overrated I decided to build my site with a flat CMS.
So I started using an Italian Open Source CMS and extending it creating plugins and also doing a fork to the project to add more stuff.
I built a tiny community around it, a mini build system and at the time I didn't know about git or other version control systems.
In 2012 I opted to use a real CMS that was following real coding standards (compared to the spaghetti code) to do more and better without starting from scratch.

Me waiting for my first Debian Ubuntu Community Conference in Italy, 2015

There was a world outside that was doing many things and available but most important they were working, and they were using a database.
In few words I discovered WordPress and with that I started freelancing, contributing to the projects and communities online (where I will find my colleagues), writing technical articles (in Italian mainly).
I also coded my first software for someone else (my father), today we call it a `bash script`.

Bash is a scripting language on Unix systems (OSX and Linux) that lets you automate software interaction.

At the time I migrated all the computers in the family store to Linux and had a simple request to

speed up the flow. Put some sheets on the scanner and automatically prepare Thunderbird and the image scanned on a new e-mail, so in 2012 I coded[30] a solution for this need, and it's still in use today in the shop.
I discovered that my English was a huge mess (today I speak only "Italish") so I resolved to improve it by studying it again, to be able to cooperate with the international side of open source projects. Submitting code, ticket reporting, discussion and proposals for let's say Mozilla or WordPress, this learning step was needed. Other communities I was contributing to at the time were KDE, Sidux/Aptosid/Siduction (my Linux distro) and many others.

Why everything started with a Linux Day

Hour: zulu 16:00 (to read as sixteen zero zero) in 22 October 2013

My first talk, 2013

Me with my 2nd laptop with "few" stickers (also wearing my flowing hairs) starting to do my first talk (Italian report)[31] at 23 years old.
I joined in the January of the same year a "Firefox OS App Days" (Italian report)[32] that was my first hackathon, and I won the first device with Firefox OS produced.
I never joined a Linux Day (in Italy is a yearly event usually organized in 80-90 cities in the last

[30] https://daniele.tech/2012/08/kde-scantoemail/
[31] https://daniele.tech/2013/10/firefoxos-utenti-sviluppatori-roma2lug-linuxday-2013/
[32] https://daniele.tech/2013/01/firefoxosappdays-a-roma-io-cero/

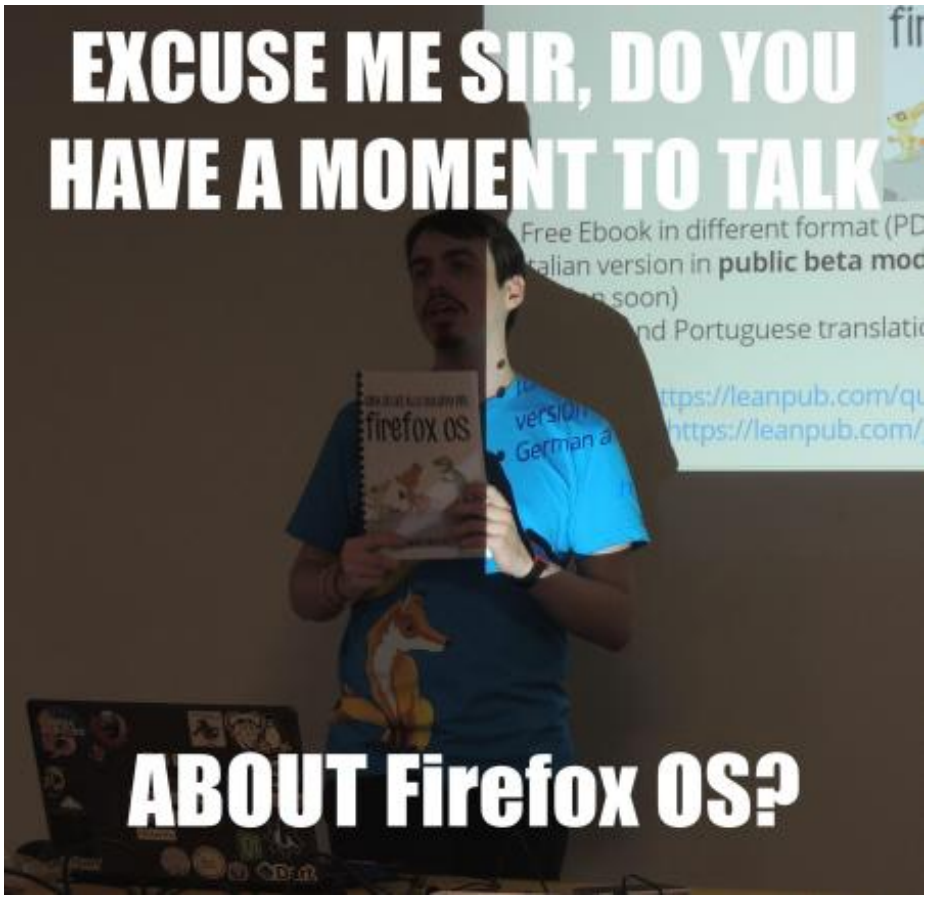
Saturday of October in the country about Open Source) but I had always wanted to go to discover and learn new things, instead of using only internet.

I did my first talk with a bit of anxiety because it was in an university, with people younger than me, but with this talk I joined the Mozilla Italia community.
Joining an event like this opened my eyes on the IT industry, because more people are interested in these things also attending in live, I wasn't alone.
This event was the first step to improve because I could discuss things with others and compare myself by talking with people to get feedback.
And, why not, have fun and network with people, that nowadays are the 2 most important parts of my role as a contributor.
For people appointed to organize community activities and events it's very important to have very good networking skills to be able to involve more people and engage them through better promotion and communication.

Mozilla, my house

Since 2013 Mozilla allowed me to participate in many higher level projects.
I met people and they supported me (even financially), so I had the chance to understand how large IT companies operate. I learned to host meetings, to manage tasks and proposals and in general many new ways of cooperating in Open Source among many.
I had an idea of how the browser and software work, their development cycle and localization problems; I learned to work in groups, to assign tasks and to research and deal with technical issues.

I also discovered what doesn't work at a higher level within IT companies, the distance they have to their users' needs (because they follow market needs instead).
I strengthened my language skills in the process.
Mozilla assigned funds to volunteers to organize or participate in events, when I joined I was one of the few experts of Firefox OS in Italy.
In short, I traveled all around Italy, visiting new places and meeting people, having the chance to better understand the needs and questions of the users of Mozilla products and of the Open Source ecosystem as a whole in general.
This traveling allowed me to build a group of friends that started to do the same, hosting demonstrations of Firefox OS, explaining how to develop for it, recruiting new people and finding new ways to engage them. Also we have improved and expanded Firefox OS Italian manual together.



The hard work of months for the Italian version of the Firefox OS development book

With the death of Firefox OS[33] I got a lot of experience in community building, mentoring, development and so on, and I kept traveling in the country also even if a little less than before.
I was giving speeches about other technologies, like addons development, Firefox web tools for developers, how to join the community, privacy and why Mozilla was important.

These travels created an "aura" around me, I was (and still I am) called many nicknames like Mozilla-Man, Mr. Firefox, Mozilla and so on. People aren't creative with nicknames after all…
At the same time I started improving my English, and in 2015 I joined Mozilla Reps program, participating in activities and discussions online at international level.

Mozilla Reps program is an international program free for everyone meant to empower local Mozilla communities (not only about Firefox but Mozilla per se or other projects like A-Frame as example).
It is a program with ~250 people from a lot of countries now (when I joined it was double that number), empowering volunteers to become better community managers, better project managers, join Mozilla discussions as NDA (Non-Disclosure Agreement) group, get funding for events and swag.

[33] https://daniele.tech/2015/05/the-firefox-os-1-x2-x-problems/

In the same period I attended my first international event the FOSDEM 2015[34], even with my messy English I was able to organize it with the help of 3 Italian friends.
Participating allowed me to meet new volunteers in person (also from Mozilla) and this helped me to meet new people (or recognize them from the nickname), slowly I understood that the Reps program was a way to help the community to do more and better.

It wasn't the time of Firefox OS anymore but there were other kind of activities to do, the real thing was that I understood better Mozilla issues.
Mozilla Foundation/Corporation is influenced by mood a lot: they start a new project (startup style), get a lifetime span of few years and later they close to work on something else and abandon everything. Even, if the project was used, important and that following the manifesto.
One of the most painful things for me was the "Web Literacy" period, when they choose to close the Firefox Student Ambassador program, to start a new one for the students and a new one called Mozilla Club.

Web literacy comprises the skills and competencies needed for reading, writing and participating on the web. It has been described as "both content and activity" – i.e., web users should not just learn about the web but also how to make their own website.
From https://en.wikipedia.org/wiki/Web_literacy[35]

This program had a very important mission, fight the digital illiteracy. For a country like Italy with a lot of it, with people who don't know how to use a computer while the smartphones was spreading it was very important.
Mozilla for many things works like a startup (I work/ed with them and I am still not liking that way of doing things), starts a lot of prototypes by needs or interests, that can be privacy (like the Encrypt campaign, when FBI wanted Apple to share the encryption keys And Codemoji), Mozilla Pulse, Maker Party, Mozilla WebMaker, Firefox Test Pilot, Firefox Hello and many others that I probably forgot.
Also the fact that Mozilla uses a lot of metrics that often are not realistic: like remove features but not understanding that they are important for a lot of users. Removing them maybe is a wrong solution, maybe you didn't promoted very well or that features is used by everyone just once every month. The cool part is when a feature is reimplemented after years in a similar way, not just because the code was ugly and was needing a refactoring but also because the feature is useful and others are doing the same. I wrote an article about how Firefox lost the browser war[36].

With employees that not always stay in the company more than 2-3 years, it is difficult to create a connection or partnership since if you have to start from scratch every time, find new references and talk with people that don't know the Mozilla experience/background and the open source philosophy.
In any (ranting) case today is always a source of discoveries and as volunteer you learn how to manage these issues, basically like how the employees themselves do. How?

[34] https://daniele.tech/2015/02/fosdem-2015-the-report/
[35] https://en.wikipedia.org/wiki/Web_literacy
[36] https://daniele.tech/2020/03/firefox-lost-the-browser-war-and-what-we-can-do/

Ignoring them, moving on new stuff with a bit of nostalgia. That way often volunteers hold the only memory of what Mozilla did because they are participating more than the people that worked there.

An important lesson is in the working environment and human, there are reasons besides the personal human/logical/financial area, things doesn't last forever except Colosseum, Pantheon, Stonehenge, Great Wall of China and Internet Explorer, Python 2 or Perl 5.
Yes, it is still around there, people are still using it and developers need to support it.

I want to close this section with links from my blog about a lot of experiences in Mozilla, I had always the habit to write a report after events or topics, and know they are a very valuable resource to understand the period and what was happening years after:

- September Mozilla Marketplace Contributor of the Month: Daniele Scasciafratte[37]
- The Firefox OS 1.x/2.x Problems[38]
- Mozilla All Hands London – I was there[39]
- ConfSL 2016 – I was there[40]
- Mozilla Tech Speaker Meetup & European Gathering in Berlin[41]
- CLSxItaly Roma 2016 – I was there[42]
- ExtStoreStats – Get stats of your browser extension[43]
- Why I am developing Browser Extensions[44]
- Analisi del sondaggio Mozilla su IoT per l'Italia[45] - Italian

[37] https://daniele.tech/2014/10/september-mozilla-marketplace-contributor-of-the-month-daniele-scasciafratte/
[38] https://daniele.tech/2015/05/the-firefox-os-1-x2-x-problems/
[39] https://daniele.tech/2016/06/mozilla-all-hands-london-was-there/
[40] https://daniele.tech/2016/06/confsl-2016-was-there/
[41] https://daniele.tech/2016/09/mozilla-tech-speaker-meetup-european-gathering-in-berlin/
[42] https://daniele.tech/2016/09/clsxitaly-roma-2016-was-there/
[43] https://daniele.tech/2017/01/get-stats-of-your-browser-extension/
[44] https://daniele.tech/2017/09/why-developing-browser-extensions/
[45] https://daniele.tech/2017/11/analisi-del-sondaggio-mozilla-su-iot/

The monument with the names of 5000 Mozillians in front of the Mozilla's office in San Francisco

I forgot to mention the huge moment of gratification, my name is on a monument with many other volunteers.

This is one of the example of how you can increase the value of what people are doing, a name is a lot of things, often in the projects is easy to limit the gratification to a "Thanks for the reporting" in a ticket but volunteers invest their time so you need to think always on how to engage them.

WordPress where my job and passions meet

To live you need the *"pagnotta"* (the food on your table), contributing to Mozilla is very enjoyable but is not enough for it.
With many years of experience inside Mozilla and at events, I started to freelance with WordPress

until opened my agency in 2015 with some friends. Let's do a step back.
I always developed with PHP and WordPress, like a lot of you may already know, is easy and has a huge community and is possible to learn and make the difference more easily.
Only the will to do and change things is required, often quite impossible within Mozilla, where volunteers are perhaps the last small part of the chain. Also, there was the programming part available that was opening more opportunities compared to Mozilla.

I started on a Google+ Italian community, hosting webinars about different topics and as technician I had more details to offer. This way I met my future colleagues, we were all moderators of this community and I started to write articles and doing support.
I was discovering this other passion after the developing, writing articles that I knew would help others from technician stuff to news. After all I had the sources, what to follow and where to find the information including what was the interesting stuff for the audience.
In the meantime I was getting trust in the WordPress community, co-organizing the WP Rome meetup and slowly also contributing to WordPress core.

Think about it, at 25 years old your first patch to WordPress was used by millions of websites around the world, was very exciting. Without considering the localization that opened my eyes in a lot of things I was ignoring.
Localization is very important for spreading software, improving awareness and helping to promote digital in your country or region.

So I did a browser extension for for Firefox/Chrome for localizers[46], contributed to tools for localizers, became a maintainer on how to manage projects and lead projects, participate in the development of other tools and so on.

[46] https://daniele.tech/2018/03/2-years-of-glotdict/

Me talking at WordCamp Europe 2017

We cannot mention the many WordCamp in Italy, so I started traveling up and down the country and up to join 3 different meetups in 3 different regions in the same week (and in 2 as a speaker).
Up to 2018/2019 when I was part of the founder committee of the first WordPress fork, ClassicPress, where I leaded the localization team and part of the core development.
Compared to Mozilla, in WordPress it is easier to receive rewards for what volunteers do, mainly because you can see your action effective instead of merely promoting other things in which you have no chance of being influential.

Also the WordPress project has issues on management, like Mozilla, and is something not easy to solve by the volunteers (again like Mozilla). WordPress as project is managed by the big company Automattic, because in the WordPress Foundation has many of their employees in the various teams as leaders, so it's easy to follow the roadmap they want: once again this is my opinion, but the fact is simple enough to confirm, look the keynote of the last WordCamp Europe/US with a roadmap that is not focused on the needs of users or communities but on their business needs.

Another point that is missing, a clear roadmap and a governance that not start from scratch but from people that already know the open source ecosystem/world. There are a lot of open source foundation to get inspiration, why we need to start from scratch and not achieve anything?
It's more simple to do that the foundation hire few people as manager of the various area that are not involved with any company and the volunteers that do what is planned and discussed all together.

Last flashback

Let's talk of 2016 when together with some friends I met online (Italian)[47] (it is easy to see the repetition of similar facts), we founded an association "Industria Italiana del Software Libero".
Our idea was to bring our great passion towards the work side, creating a group of professionals who use open source software on the professional side and contribute to the same.
A great idea that today has failed to explode as we thought, also because we are professionals who have to work and it is difficult to move forward.

[47] https://daniele.tech/2016/03/industria-italiana-del-software-libero-la-nascita/

L'innovazione

La sfida dei mille

Tanti italiani hanno risposto dall'estero all'appello del team di Diego Piacentini Obiettivo: migliorare il software dello Stato e condividerlo con enti, comuni e regioni

Le storie dei nerd in campo per cambiare volto all'Italia

JAIME D'ALESSANDRO

ROMA. I mille nerd che vogliono salvare l'Italia. Quelli che in 48 ore dalla prima chiamata alle armi hanno risposto. A differenza di 157 anni fa, questi non son partiti da Quarto. Arrivano da Dublino, Londra, Udine, Trento, Berlino, Rieti, Roma, Milano. E alcuni di loro lavorano per multinazionali note, da Facebook a Microsoft fino a Slack. Un esercito che sta entrando nella comunità di Developers Italia, il sito per gli sviluppatori che vogliono mettere a disposizione la propria conoscenza gratuitamente per migliorare la Pubblica Amministrazione. L'appello è dell'Agenzia per l'Italia Digitale (Agid) e del Team per la Trasformazione Digitale guidato da Diego Piacentini. Era difficile prevedere una adesione simile. Molti se ne sono andati per necessità dal nostro Paese, altri diffidano delle istituzioni, tutti hanno una pessima opinione della burocrazia. E invece eccoli rimboccarsi le maniche, dandosi da fare sulle aree più critiche. Già nelle prime ore dall'apertura del sito, hanno cominciato a scrivere linee di codice correggendo e aggiungendo i pezzi che mancano all'Italia digitale. Ed è tutto software libero, trasparente, applicabile, replicabile da comuni, regioni, enti.

«È un modo moderno per migliorare le piattaforme e i programmi che vengono immessi man mano nella Pubblica Amministrazione», racconta lo stesso Diego Piacentini, ex vicepresidente di Amazon. «Ogni volta che si lancia un servizio, come il Sistema pubblico di identità digitale (Spid), i pagamenti digitali dei cittadini (PagoPA), le fatturazioni verso la pubblica amministrazione (FatturaPA), è sempre una prima versione: inevitabilmente ha al suo interno errori e problemi di vario tipo. Come ogni progetto deve e può essere migliorato se lo si vuole applicare. E lo stiamo facendo grazie alla comunità di sviluppatori. Quando abbiamo presentato l'iniziativa, lo stesso giorno, uno di loro ad esempio ha tradotto in inglese tutte le pagine del sito».

L'annuncio è stato fatto pochi giorni fa in un evento di Codemotion, ciclo di conferenze europee con una rete di 35 mila sviluppatori, ospitato al piano terra della facoltà di Ingegneria a Roma Tre. Computer portatili, magliette con loghi storici dell'elettronica di consumo, età media attorni ai trent'anni. La Pubblica Amministrazione d'ora in avanti potrebbe passare per appuntamenti simili. E avere il volto di chi li frequenta. Bisogna solo capire se sia il segno di una inadeguatezza tecnica oggi difficile da giustificare da parte del pubblico o una vittoria perché finalmente si eviteranno lentezze e lungaggini. In attesa di decidere se il bicchiere è mezzo pieno o mezzo vuoto, una precisazione: non immaginatevi il ragazzino di *WarGames* alle prese con il sistema di difesa missilistico. Qui si lavora su "dettagli" meno affascinanti, ma che impediscono che le cose funzionino.

Mettiamo ad esempio che un comune voglia inserire sul suo sito il tasto con il collegamento alla propria pagina Facebook. Il social network fornisce tutte le indicazioni necessarie per farlo, la spesa da sostenere per il comune quindi è bassa essendo una pratica che non ha nulla di esoterico. Se però si tenta di fare la stessa cosa con il Sistema pubblico di identità digitale (Spid), per consentire ai cittadini di accreditarsi con un click, manca qualsiasi documentazione. Il risultato è che l'ente sarà costretto a spendere una cifra dieci volte superiore a quella necessaria per attivare un collegamento a Twitter. Di più: l'esborso di denaro pubblico per risolvere problemi del genere fino a ieri veniva ripetuto enne volte da tutte le amministrazioni invece di condividere fra loro la soluzione. A Milano hanno appena aperto il Fascicolo del cittadino: un luogo digitale dove sono raccolte tutte le pratiche che riguardano gli abitanti, dalla tassa sulla raccolta dei rifiuti, alle multe, ai servizi scolastici e in futuro le informazioni legate alla propria auto o moto. Ora quell'esperienza potrà essere adottata, anzi adattata, da altri comuni a costi ridotti, grazie all'opera dei mille.

> L'ex n.2 Amazon: "Come ogni progetto, può e deve essere migliorato. Dalla comunità di sviluppatori"

«La realtà è semplice: prima le risorse non c'erano per fare queste cose e anche se qualcuno avesse voluto dare una mano, non avrebbe potuto farlo», spiega Giovanni Bajo, 37 anni, a capo delle relazioni con gli sviluppatori del Team Digitale. «Ora ha un luogo dove fare proposte e pubblicare linee di codice che verranno controllate da tutti e messe in circolo in tempo reale. Non mi sembra una cosa da poco. Speriamo di avere i primi risultati importanti nel giro di qualche mese». Già, speriamo.

DAVIDE GUERRI. 40 ANNI, BRISTOL

"Lavorando a Facebook aiuterò il mio paese"

È nato a Roma, ma da tre anni anni e mezzo vive a Bristol dove lavora come production engineer a Facebook. «La trovo un'iniziativa fantastica», commenta. «Applicare un modello di sviluppo aperto ha dei benefici enormi sia dal punto di vista della qualità finale del prodotto, sia perché si rende trasparente una realtà che è sempre stata autoreferenziale. È una mentalità affermata all'estero. Io ho già dato un modestissimo contributo e appena trovo un po' di tempo libero collaborerò ancora. Perché sono emigrato in Inghilterra? Non sono riuscito a trovare un ruolo interessante in Italia. Il problema è stata sicuramente la crisi economica. E poi a Roma tutto ruota intorno alla PA. Che in quel momento non investiva più nulla... Magari adesso cambierà qualcosa».

GUIDO IAQUINTI. 26 ANNI, DUBLINO

"Partiti col piede giusto in 2 anni primo bilancio"

Originario di Imperia, da quasi cinque anni ha lasciato l'Italia. Prima a Berlino, ora a Dublino. Era inizialmente nella divisione Cloud and Enterprise di Microsoft, che ha come clienti anche tanti enti governativi. Ma da quasi un anno è responsabile delle technical operations a Slack, l'app per la comunicazione collaborativa e aziendale. «Developers Italia mi sembra un'ottima iniziativa e spero prenda sempre più piede fra gli sviluppatori italiani. Per il momento ho contribuito rispondendo ad alcune richieste specifiche fatte dal team in attesa che rilascino qualche progetto di ampio respiro. Attualmente quelli davvero grossi sono solo un paio. Per un bilancio penso sia giusto attendere la fine del loro mandato, fra 18/24 mesi. Ma sono partiti con il piede giusto».

DANIELE ESPOSTI. 37 ANNI, LONDRA

"Una burocrazia antica serve la svolta digitale"

«Fino ad ora la Pubblica Amministrazione ha usato la tecnologia come effetto collaterale più che come strumento principale per fornire i servizi al cittadino», racconta Esposti, Senior Data Platform Engineer in Badoo, il sito inglese di dating. Ha lavorato in Texas prima di trasferirsi in Inghilterra ed è stato lui a tradurre in inglese il sito di Developers Italia in appena due ore. «Questo mi sembra l'avvio di un processo mentale», spiega. «Trasformare una burocrazia ottocentesca in un fornitore di servizi al cittadino alla pari delle aziende digitali odierne. Sono conscio che per attuare questa trasformazione gli ostacoli sono tanti, bisogna cambiare le abitudini consolidate da decenni. Richiederà una quantità di energie e risorse notevole. Ma è questo a mio parere il vero obiettivo del progetto».

RIGEL DI SCALA. 39 ANNI, DUBLINO

"Accesso semplificato e il cittadino al centro"

È nato e cresciuto a Milano, ma dal 2012 vive in Irlanda con la sua compagna. Ha curato i servizi digitali di cinque Stati federali negli Usa quando lavorava alla Propylon. Dai sistemi di votazione a quelli per l'approvazione delle leggi. Nella pubblica amministrazione americana ha messo mano. «L'iniziativa di aprire al software libero e agli sviluppatori non è una novità assoluta, ma è avanti rispetto a tanti, iniziando dagli Usa. Ed è ben più avanti degli strumenti usati dal Movimento 5 Stelle che non hanno nulla di libero e trasparente. Insomma, l'approccio è interessante: semplificazione dell'accesso, coerenza fra i vari servizi e cittadino al centro. "Il successo, in Italia, è sempre visto come un furto a danno dei mediocri", scriveva Michele Serra. È ora di cambiare».

CLAUDIO CICALI. 49 ANNI, BERLINO

"Basta con le piattaforme che non parlano tra loro"

Toscano di origine, della zona di Pisa, scappato dall'Italia perché «troppo complicata per un programmatore indipendente». Oggi lavora come capo dello sviluppo web in un'azienda chiamata N26. «Nella mia passata esperienza professionale ho avuto a che fare con le Ferrovie Nord Milano, Banca d'Italia, Poste Italiante... e so benissimo quali sono i problemi che il Team Digitale si trova ad affrontare: difficoltà di integrazione tra servizi che parlano lingue diverse, contrasti da parte di persone che pensano di venir sorpassate e che invece vorrebbero mantenere il controllo, conflitti politici (*do ut des*) e via discorrendo. Sono straconvinto che l'idea di rendere tutto aperto e trasparente sia l'unico modo di operare in quel settore. Dare il mio contributo è il minimo».

DANIELE SCASCIAFRATTE. 26 ANNI, RIETI

"Creare sistemi aperti è avere senso civico"

Vive di open source, di software aperto che viene migliorato in maniera collaborativa. E da quasi dieci anni fa parte sia dell'organizzazione no-profit di Mozilla sia di quella di Wordpress. «Mi piace: oltre ad essere un progetto per una volta italiano, apprezzo l'idea di una comunità di italiani che si occupa di software aperto per i cittadini. Collaborare significa avere senso civico. Ho solo qualche dubbio: possibile servisse un altro team oltre ad Agid (Agenzia per l'Italia Digitale, *ndr.*)? Evidentemente era necessario che qualcun altro risolvesse il problema. Da sviluppatore che si è letto tutto il materiale, spero che il Team Digitale prosegua con questo modo di fare. Un approccio che alla Pubblica Amministrazione manca decisamente».

Article from Italian newspaper with also my interview about the Italian Digital team

I say it because in 2018 I became the president of the association and after a few years I can see the issues in a world that transform itself too much quickly.

Let's think a moment together, in 2019 the open source world is not as strange and dark as 5 or 10 years before.
The world is paying more and more attention to the opportunities of these technologies (except the Italian government in the various cities) but I see the big problem: to contribute.
This book was written because I see a lot of young people and newcomers that ask how to do something, how to start or how it works. Because it is more important to be able to participate and contribute than to be mere users of the product for them.

Addendum (third edition)

Writing a new edition required two years with the Covid pandemic that broke the flow of ideas generated during FOSS live events, but I didn't stop gathering resources and doing Open Source.

Me with the Public Money Public Code T-Shirt at camera dei Deputati

On February just few weeks before the Covid I joined an event where I was the representative about the Open Source topics in the Italian Depute's Chamber[48].

On December 2020, I started my podcast in *Italian* about weekly updates from the Open Source world with a bit of cyber security and technology updates with my opinions. As today with more than 90 episodes it was an experiment that worked with more 220 followers (on Spreaker, Funkwhale and Youtube I can get those numbers but not on the other services). Every week I gather an average of 34 links to discuss in less of 30 minutes. I wasn't thinking about all this interest in these topics as it is very specific, but looking on the people who reach me to share feedback or links seems growing.

On January 2021, I was added also on the `about:credits` page in all the Mozilla Products like Firefox, for my contributions in the past years. At same time the "Industria Italiana del Software Libero"

[48] https://daniele.tech/2020/08/etica-open-source-riuso-e-software-libero-versione-originale/

association was closed because there was no action from the members on doing the usual things, it was a good experiment on how to manage this kind of things.

On April 2021, I was elected in the Italian Linux Society[49] Association council and I will be part of this for the next 3 years. I was happy to join it because I wanted to give back to my country after doing international stuff in various communities. As association, we help all the Italian Linux User groups and promoting Open Source in Italy and we did various things in 2021 like 9000 € in donation to FOSS projects[50] or I organized the Merge-IT[51] conference online with over 12 Italian Open Source communities or about Digital Rights to discuss various topics.
I put my experience like in these ebooks to improve various websites we offer like Planet.Linux.it[52] with a new calendar and ICS/RSS feeds divided as national or by regions to find the nearest events to you about open source, so not just LUGs but also from meetup and national communities.
Or a whole series of tiny/easy guides to contribute to Open Source projects from your smartphone/-computer in 5 minutes[53], not just using an open alternative app.

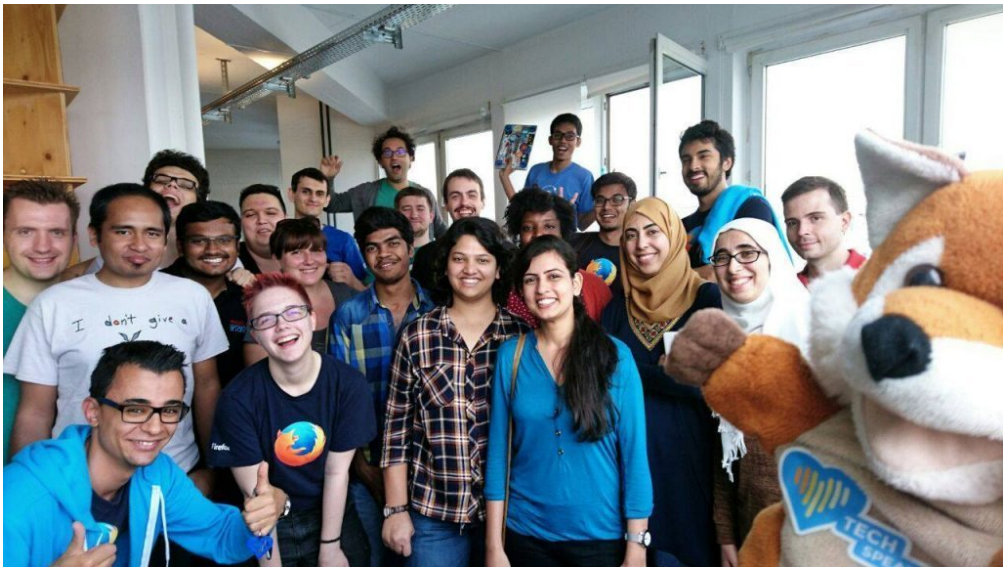
In April 2022, I joined the Italian community behind /r/place, and it was an interesting experience[54] in something that last few days but purely online.

[49] https://www.ils.org/
[50] https://www.ils.org/2022/01/01/2021-con-ils.html
[51] https://merge-it.net/
[52] https://planet.linux.it/eventi/
[53] https://www.linux.it/todo/attivita/
[54] https://daniele.tech/2022/04/the-r-place-story-from-the-italian-view-and-from-bots/

Conclusion

Open source allowed me to:

- Gain experience despite the fact I never joined university
- Improve English (this book was written half in Italian and half in English and reviewed from other people, thanks!)
- Improve my relationship skills (no one thinks I was a shy guy)
- Improve the public speaking skills
- Understand how to promote stuff
- How to split problems
- How to manage a working group
- Know what is needed to organize an event
- Understand which are the requirements to keep a community alive
- Compare with others and see my limits/problems
- How to ask to… ask and not getting troubles for it
- The various aspects of a project (not only about a software)
- Learn how to develop, the right way
- Learn how to listen and when it is time to shut up
- How to propose stuff



Mozilla TechSpeakers at the 2016 Meetup in Berlin

A lot of very useful things for work but also for personal life, the fact is I did everything for… fun!

Your turn I want you repeat the same flow of thoughts as I did on writing this chapter. Take 2 analogic tools, a pen and a paper and write on that:

- The first time that you participated in an open source project
- How many years ago
- List all the things you learned and mark the most important lessons of your life

If you never joined an open source project:

- What do you want to replicate of this story
- Where you see yourself in this story
- What is your biggest target (also non-realistic)
- What is the first step you want to do
- What you want to learn

There are also other FOSS story around with less photos but more clear learnings (maybe) like this one[55].

[55] https://blog.burntsushi.net/foss/

My (or your new) philosophy

If you are still reading this pages it means you found them interesting/useful, or you think first chapters do make sense..
Let's speak about the Philosophy now, I don't want to cover the four freedoms in depth here (for there are articles covering this topic already), but you will find a brief recap below.

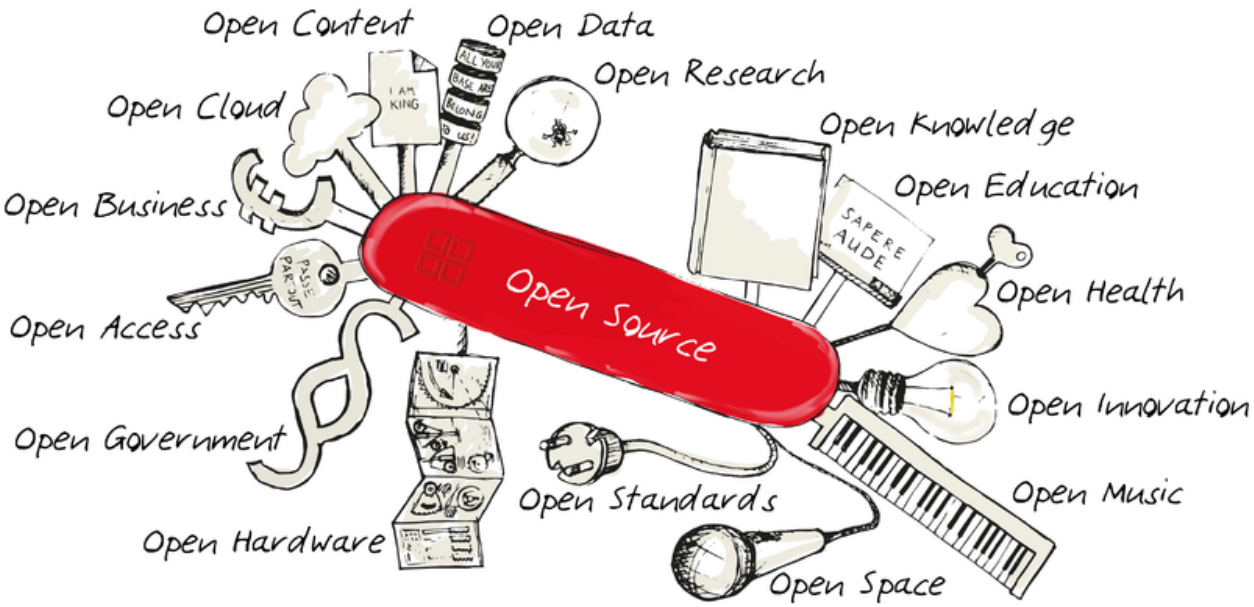
The 4 freedoms of Open Source software/projects

- The freedom to run the program as you wish, for any purpose (freedom 0).
- The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help others (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give to the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

From: https://www.gnu.org/philosophy/free-sw.en.html[56]

This 4 freedoms created the philosophy and the open source model.

[56] https://www.gnu.org/philosophy/free-sw.en.html



Little recap about open philosophy

Open source model examples: Open Hardware (Arduino or Raspberry Pi except for CPU), Open Content (Creative Commons license), Open Knowledge (Wikipedia or Project Gutenberg), Open Data (OpenStreetMap), Open Access (Open Document File formats)

We need to exclude the biggest metropolitan legend about this topic: open source doesn't mean that it is free of charge (also if everyone use OSS, including enterprise). This point is very difficult to understand for newcomers and if you have troubles, well check it out reading the rest of this chapter.
Another important benefit of Open Source is that you are getting in addition to the product a bonus, the access to the various pieces that build it. It is like getting a pizza but also the recipe with the various information for any ingredient and so on. For a non-technical person this can be useless but when you need to study how works or implement based on the requirements of your customer is very important because let you to do a full analysis.

Good artists copy; great artists steal. From: https://quoteinvestigator.com/2013/03/06/artists-steal/[57] In the IT world there is another version: "Reinventing the wheel" but I prefer to add also "What is not understood is not possessed" of Johann Von Goethe to help you understand the philosophy.

The seven stages of Grief is a way to express the various steps of a loss, anyway someone converted as Seven stages of OpenNess[58] that can be used as mnemonic to understand how the Open Source

[57] https://quoteinvestigator.com/2013/03/06/artists-steal/
[58] https://coiled.io/blog/stages-of-openness.html

process can change your view on doing and understand FOSS (read this link before this chapter if you want to see other opinions). Another idea that surprised me is the boy-scout rule “Always leave the campground cleaner than you found it”, that is part of what I do in Open Source and this book is part of.

So Open Source != Free Software, this means that open source is a subset or a minimal definition of the goals of a project compared to Free Software. This happen about the *free* in Free Software because as we said open source means that the code is available but what you can do with that depends on the license that can be Free Software, or that you need to pay to access it as example. There are tons of pages that explain the difference between the various license and why but often they are very complicated, so get ready on studying them[59].
Another opinion is by company side when the license topic is on fire, like dual licensing, CLA, abuses or license’s restrictiveness but I don’t want to talk about those stuff as it is the most common discussion in this world and there are people who explain a lot better than me, Open source licensing for supervillains[60].
Yes, you need to study a bit because it easy to get a license conflict because you like to code but you don’t care of the legal part[61].

Do ut des

Do ut des is a Latin expression that means *give and take* but we can openly localize as “something is given so that something may be received in return”. In English often this can be understood as *giving back*, for me this is not enough to explain it the idea behind. When we need a software or a service the first step we do is to look it up on the net. Then we download/register, use/consume and then move on with our day.
We are just consumers, and often we don’t know the costs involved in developing such software/service.
When using free web services or apps, many people don’t ask “how can this service continue to move on if it’s free?”, but it’s a question we should ask ourselves.
How Google/Facebook/Whatsapp/Youtube/Gmail etc still offer their services?

[59] https://blog.graphqleditor.com/software-licensing-cheat-sheet/
[60] https://offlinemark.com/2021/01/22/open-source-licensing-for-supervillains/
[61] https://arkadiuszkondas.com/dmca-php-ml-and-copyright-boundaries/

What they do with your data? Fosdem 2017

You are the currency, using it and leaving information in your account and surfing the web, with this information they can show to you perfect ads or discover interests your kind of person has to use it for something else.
In the open source world this doesn't exist, a service works because some people behind it use their free time maintain it (yes, there are people paid for that sometimes). When you are using these technologies you are accessing to solutions that for you are free but behind there are many people that works to let you use them freely (classic example Wikipedia).

Why everything starts in the open source ecosystem? Because a user sees a problem, something missing, text written badly or with typos, incomplete localization and so on.
Basically the contribution starts when something doesn't sound so good and the issue is quite clear. When the problem is noticed instead of doing a useless comment online or a ranting at the pub/bar, there is an action with a report and the problem is/will be gone. This is the contributor, that enforces the contributor mode (next chapter) without noticing it just because OSS require commitment.
This report is the first step to contribute to the project, because behind there are people that are not perfect and that can miss something.
The society doesn't understand all of these things, maybe the 4 freedoms are more easy to accept but the commitment on something that is from the IT world is very strange. We need to help the society to understand why this commitment is important, like the civic hacking or other reality that in the IT world act to improve the quality of life using the technology.

i> Civic hacking, “enhances the relationship between the people and government with software for communications, decision-making, service delivery, and political process.” Wikipedia[62]

The real story

Another way to see this topic is leave everything better as you found it originally.

From a company view, we need to think about another thing, **production** and **consumption** are 2 different things that require different people and skills. The idea of this book is to help address the various side of the Open Source world because a *Maintainer* cannot do everything alone.
“Users need open source projects, but open source projects do not need users[63]” this quote is from a story from an OSS project that has a company behind that reminds to everyone that a project doesn’t mean that need to be suitable for everyone, also if at same time can be transparent etc.

The maintainer is a role that is a project manager, support, developer, documentation and so on. Basically everything that start from promotion to keep the project alive, and if there is growth can share these duties to other people. It is a time consuming role and require many skills:

- Improve documentation based on your experience

[62] https://en.wikipedia.org/wiki/Civic_technology
[63] https://matt.life/writing/the-asymmetry-of-open-source

- Be open to activities for other users
- Propose and try new approaches
- Be honest with/care others
- Define roadmap
- Assign tasks wisely
- Janitor tasks

And so on, but you can do those tasks and help the projects you love and improve the value over all.

In a company, this exchange of value by contributors or the future of a project is a completely different thing, for example a project can be released for branding promotion but ignoring proposed changes[64]. It is known that company usually hire people who contributed to specific projects that they use or are focused on the area they need skilled people.
This company projects often have difficulties on managing the open source community if they are not structured to handle external involvement, but this doesn't mean that you can't contribute to them. When you are working in these cases it is required to think in a different and more complex way, Open source is not just the 4 freedoms but also a world around visibility, promotion and business (that are not bad things) and not just transparency.

From the Principle side, we can talk about the Robustness Principle[65] in OSS. That can be translated in "be conservative in what you send, be liberal in what you accept", you are open to the world to look at what you are doing, but you don't want the involvement of everyone. This is a non-sense as per the 4 freedoms but after all you don't want trolls, bad people and so on. To move on a project you need to think in the real world where you have to chose what is better for your project to keep going on.

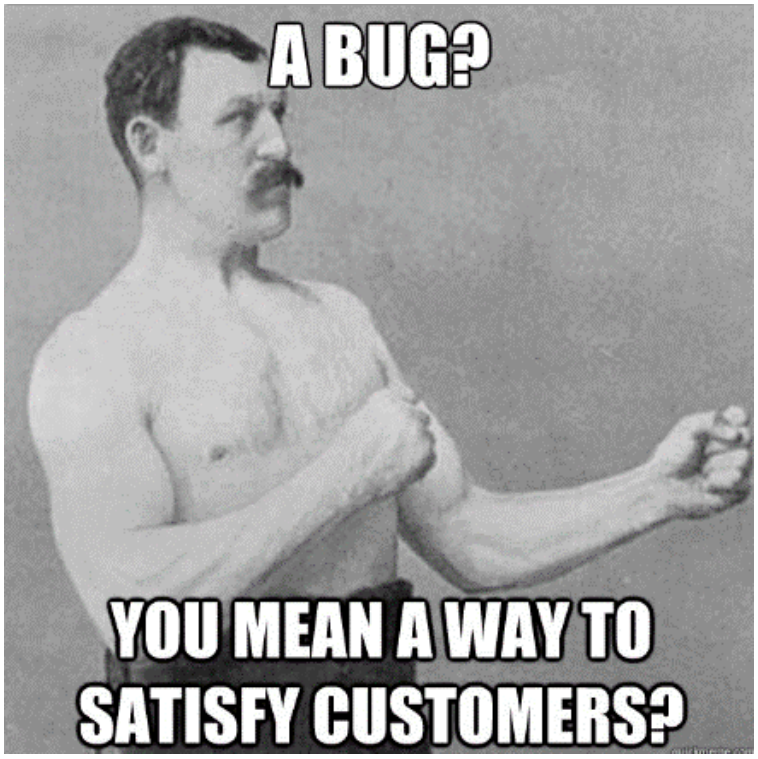
From a learning person or a student or someone who is looking to get experience or fill the famous curriculum, it is different. You read my biography in chapter 1 and you saw that I made myself on my own with open source but can be difficult to show this expertise outside this world. Requires personal branding but also documenting somewhere what you are doing it as you are exchanging your time on contributions to grow your knowledge and experience. There are project like Google Summer of Code[66] where students can get paid (if succeed) to contribute for a project on a specific task, and they will get a mentor or Outreachy[67] that "provides internships to people subject to systemic bias and impacted by underrepresentation in the technical industry where they are living".
Anyway it's not something new that there are projects that are opened to university/college students as there is more attention today on the positive things that they can learn on this[68] and if you are reading this book probably you have already some hints.

The first interaction of an user is an opportunity to get a new contributor, so as community manager, maintainers, contributors etc is important that we offer the best experience, the best gratification we can give but also the best value both the parties can get from this action. In this book we will see a lot of suggestions, ideas and process to achieve it starting from the… reporting!

[64] https://miles.land/posts/corporate-open-source/
[65] https://en.wikipedia.org/wiki/Robustness_principle
[66] https://summerofcode.withgoogle.com/
[67] https://www.outreachy.org/
[68] https://cacm.acm.org/magazines/2021/7/253459-why-computing-students-should-contribute-to-open-source-software-projects/fulltext

Reporting

It seems a joke, but it's not; the real culprit is complaints have to be useful to reach a constructive goal. Usually people stop at the very first step, without acting to make it a constructive reality. This only creates a bad mood and the effect is the opposite of the desired one.
Give the maximum of information to developers, in order for them to research the issue, replicate it (that is a very important step) and create automated tests for it. By only complaining you obtain nothing, you just get madder than before.
The ones able to report issues the correct way, and with much information, show a different knowledge level than others; that is a very helpful skill set to have during daily operations.



How to see the problems

Enterprises apply that way of thinking, but an open philosophy project cannot have this limits.
You can do your part to change the world, and after some time see the problem solved thanks to your intervention! At the same time all other users will have same benefit.
Little step for a man, huge step for humanity as a whole.

When working with public domain resources, there shouldn't be any *omerta*, for it's against the philosophy of the open source projects itself. Everybody can do her/his own part, even with little time available or doing just little things. A puzzle is made by thousands of little pieces, and everyone can contribute competing it.
Nothing is more distant from civics (that in Italy is very poorly understood, even if now it's mandatory to teach it in public schools).

Open Source is adorable

There is also another kind of reporting, the one that can help your community on track what is happening or just gathering the common questions and write some documentation. Those are more tasks for a community manager and will be defined in other chapters on the book.

How to do a ticket

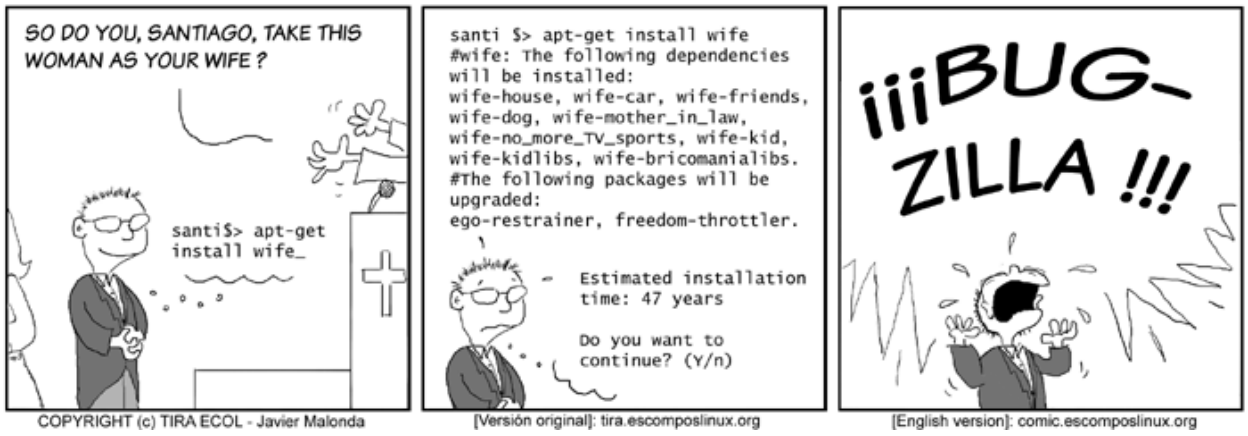
Who manages a project often **doesn't have a lot of time** so you need to give the most possible of useful information (and not ranting). Usually *giving some example* or user case helps avoid a shallow reject and will force to read the ticket with an analysis of the details. Basically the own vision of the problem or of the feature request is very important to give the context or user case/story. Also one ticket should address only a single topic to avoid confusions and lost of information. For example, they reply to your ticket after 3 months asking for more details but you remember nothing. If you wrote them easy-to-read information then your request will be processed quickly there isn't the need for further exchanges.
The writing style should be friendly. No one wants to read rants, even if constructive ones. Also, often issues may be caused by a wrong usage of the product or by something that went unnoticed. This is an useful approach for support centers too, behaving in a friendly way will encourage the support reps to act friendly towards you and be more helpful than usual.
In doubt cases, it is advisable to end the message with some polite expressions like the ones below:

- It's my first time using the project, so I am not sure…
- I am not sure to use it the right way…

- Am I doing something wrong, or is my way of handling the product correct?

This language trick allows you to make requests in a friendly way (without efforts), shows that you *desire to discover* more and makes replying to you more simple to the receivers.
Honestly, I saw this way of expressing working wonders many times, in case of misunderstanding you may also explain that you have some issues with the language, or you may point out you are not an expert in the field, this also helps in toning down noise during discussions and engages other participants empowering them to express and explain themselves better.

Don't forget, when you need to pose a lot of queries, it's easy for people to find you annoying, thus they behave unwelcomingly. Their time is precious, so for them collecting information and basic understanding of the issue in a timely manner is a culprit. That's why I advise you to select a friendly and informal writing style that allows you to achieve more at first approach.



How to reply to a ticket

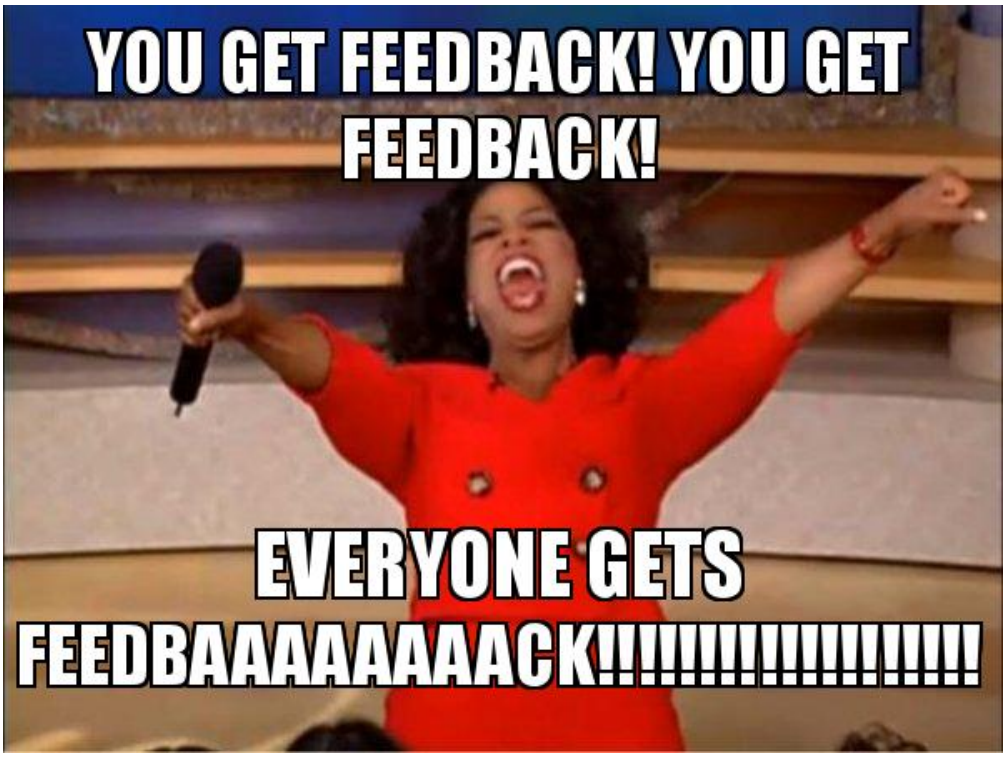
If the project is our, how we should reply? Or when we contribute to the project validating the tickets (called triage) how we should reply? Quite simple, follow the same rules, avoiding the part of the non-expert of course!
Ask for more info, verify the request or reference to the code in the project answer (this one helps others to join and ease the task for you, basically onboarding new people) shows your own professionality and availability.
Anyway, it's important to document also within the closing message of the ticket. The request in itself is valuable for the person invested time in redacting it and it's very awful seeing them closed with a bare "It's not our priority".

A quickly answer often is highly appreciated because means that you are following what is happening in the project and to be very involved/interested in what is happening and fix the issue. Often I propose ways to accomplish my feature request or bug fix, that way I can start a discussion (and be part of it) to involve others and encourage them to do the same.

Concluding, don't forget to take responsibility in case you are the culprit (by not reading the manual, due to misconfigurations and, so on) because you have to be honest; document the issue for someone else and simplify the job of the project manager.

The virtuosity of reports

Let's think for a moment to all the rants or requests we receive every day, to how much better they would be if submitted in a **constructive way**.



There are 2 kind of people: who want a feedback and who is lying

Gorgeous, isn't it? Let's take the first difficult step ourselves, let's be ready to take it because we know how a ticket is powerful, and we know how to take it, so we can spread it to all the world and in our daily life, family, job and so on.

On a personal level, it helps thinking in a constructive way, it helps in lifting depression and avoiding being angry with other people around us (I guess that moment before the proverbial "morning coffee" is an exception to that); it helps in making us more optimistic, it makes us grow better, when the request is well received and we wait like a child on Christmas Eve for the outcome.
In life, when something is not working we need to be listened, but the only way to achieve that is to be constructive and available because this speeds the process up a lot, by improving communication.

Obviously, this isn't a fool-proof method, but makes us trustable, it transforms the environment into one where interactions are productive/pleasurable/easy (when there is a shortage of time making

things hectic, too!). Communicating that way also creates an "history" that may come useful in future times ahead as a reference.

Another point if in a OSS project is not possible to do questions it isn't a real OSS project. Another view can be you can do questions but there are some choices that for [1-2-3-etc] reasons is not possible to ask again.

How to evolve it to the next Pokemon

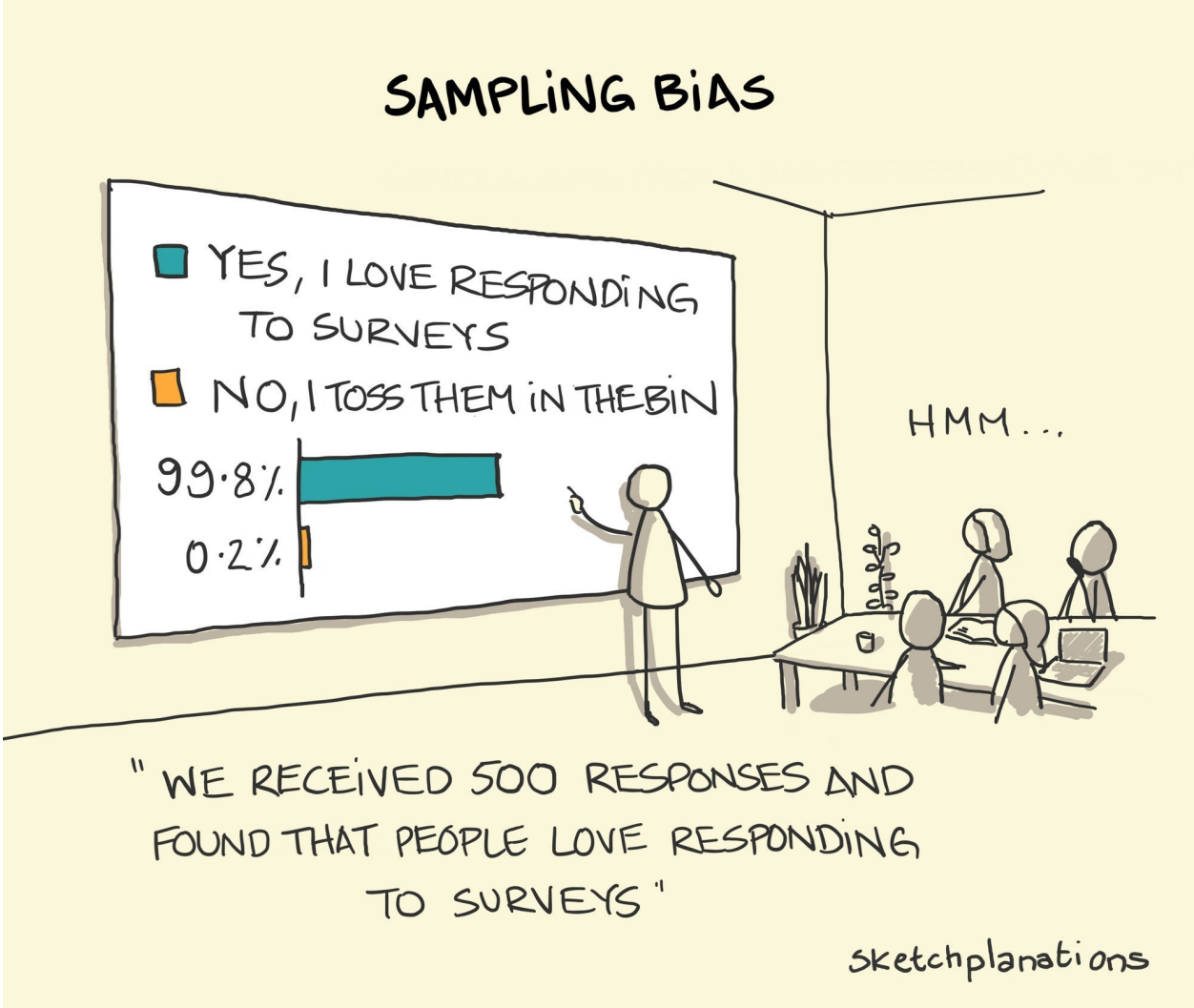
To open some tickets (or participate into existing ones) is very important, it allows you to get involved at a very low level and to join the party.
For a lot of people the real question is: how I can live with that? The famous "pagnotta" already mentioned. Well, RedHat is the most profit company about Open Source (now part of IBM) released a report of 2019-4[69] about the enterprise world revolution with FOSS.
The main points:

- Modernized the companies
- Improved the security of their infrastructure
- Lower total costs
- Integration with others platform they use
- Ability to customize the software

Those are just few hints from this reports that show how can be powerful embrace this philosophy and solutions. For IT companies also open new doors for business like this Tidelift's report explains to getting more developers[70].

Let's consider that a way to start a relationship in the way others expect this to happen, like the fish, when we move it from his tank to a new one, we make sure the temperature of the tanks is consistent and the environment is similar, to avoid it any shock whatsoever.
The next chapter approach a bit how to improve the quality of your learnings with this new philosophy. Now that you joined, what's next?
It's up to you, doing the triage you learned more things on how the project is organized, who does what, who works on what, where to find the right people to help you without being overwhelmed by the hugeness of the community, you learned to estimate response time and also the actions you can take from there.
Next chapter is about how to maximize the Contributor Mode.
I call it so because I am a nerd, and probably because eating so much candies, chocolate, liquirice, fruits and the like, I get to drink not enough coffee.

[69] https://www.linuxfoundation.org/resources/open-source-guides/winding-down-an-open-source-project/
[70] https://tidelift.com/subscription/managed-open-source-survey



The survey Bias

For this reason, is important to consider the Bias when you are doing evaluation on FOSS, as there are tons of enthusiasts, but they are just part of your users, so they can alter your numbers and ideas very quickly. Their opinion is important but is important to have a full view.

How to manage a new project or a legacy one

This is an hot topic today with a lot of technical debts around, old projects released open source or to the public (two different things)[71] or a project started in the company internally now is ready to be published to the OSS world. There is also the case of an old project release with an OSI license with some changes[72] with discussions about the declaration of opensourceness. There are tons of article about how to manage a legacy project/codebase but I don't want to talk about those.

[71] http://distractionware.com/blog/2020/01/vvvvvv-is-now-open-source/
[72] https://forum.defold.com/t/we-are-finally-sharing-the-defold-source-code/65177

The real point is that if you are starting a project is important that the code is ready from the start to be released as public just because the project rules, tickets, commits, and so on will be free of badwords (I know what happens), non-english stuff and documentation etc is ready to be released. Take as example if you are hardcoding tokens and other things that is better to avoid in your commit history, if there are issues in non-englush languages mentioned in the code that the people doesn't have access to (originally LibreOffice code had code comments in German as example from the StarOffice origins).

"A security token is a peripheral device used to gain access to an electronically restricted resource." Wikipedia[73]. In IT there are also token generated by apps or websites used for authentication.

The transparency of the OSS management can benefit the quality of the project itself also if internal and to new people that will work on that.

For legacy project instead it is important to define the license, how to setup a dev environment (if it is possible), give all the information that are available and that can be helpful to run your tool. Don't be like this (source code of XIII game that I like so much, also the comic)[74] that doesn't include any readme or documentation.

[73] https://en.wikipedia.org/wiki/Security_token
[74] https://github.com/Ch0wW/xiii_unrealscript

Conclusion

I hope that now it's clear why Philosophy is so important: not only it differentiate us from our competitors (the non-open projects) but it's for us a path to follow.
Open source without understanding the philosophy or way it should be open or managed openly, is only another proprietary project hidden by the code availability. Again a project without a license is not an open source project.
The real question could be *Why you should join a OSS project?*, let's see some points:

- Improve your knowledge in different areas
- Visibility on your field
- You can find a job or new employees
- You do something also for others
- You will see some effect of what you do in other people or areas
- Learn the rules to work with other people
- Learn how to verify the problems/issues reported
- Fight against demotivational critics
- Helps others also with code
- Learn when avoid Jargon language
- Find people similar to you
- Find mentors

I want to focus again (and I will do often in the next pages) how much an Open Source project is not just the software, but without it is empty.
A real example is the Jekyll case[75], where there is this list of what is required today for an Open Source project:

- Engagement on Twitter (or where your community is)
- Official Discord chat room (or where your community is)
- Public roadmap
- Predictable release cycles
- Welcoming community involvement in shaping new features and tackling technical debt
- Cultivating working relationships with wider ecosystems (like the techstack communities)

This list shows us how much is changed when everything started in the 70s with Unix, but at same time how many skills are required to manage a project with a community. Of course if you are

[75] https://www.bridgetownrb.com/future/rip-jekyll/

looking for a first contribution it is more simple but is important to know that is not just the GitHub's Pull Request page.

Before the last words of this chapter I want to leave this news, 'The US military wants to understand the most important software on Earth'[76]. As today cybersecurity but also awareness about digital security is day after day more important, the defense department of the USA is trying to understand what this means for the whole internet and their own infrastructure.
The result of this question:

- Open Source projects/communities need a higher level of care and respect
- You can't have thrust on the codebase just because it is OSS
- Code can be written by sanctioned entities (from specific governments)
- The bus factor is limitating (the whole project fall apart if those number of persons get hit by a bus)
- Proprietary software are built with 70% of OSS

Those are some examples that can help you to find opportunities for you but we will see better in the other chapters.
I am leaving to you some examples from the internet about how other people explained their first contribution:

- First commit to KDE[77]
- My contributions to GNOME as a non-coder and how you can too![78]
- Why I help people with PHP RFCs[79]
- Interview with PHP Core Developers[80]

Don't be just a OSS promoter but a OSS contributor!

[76] https://www.technologyreview.com/2022/07/14/1055894/us-military-sofware-linux-kernel-open-source/
[77] https://christianchristiansen.net/log/kde-first-commit.html
[78] https://jatan.blog/2020/04/11/my-contributions-to-gnome-and-how-you-can-too/
[79] http://blog.basereality.com/blog/25/Why_I_help_people_with_PHP_RFCs
[80] https://thephp.foundation/blog/2022/05/06/interview-with-core-developers/

Contributor Mode

I call “Contributor Mode” that moment I use to code patches, localize or organize events (no specific key combo here – I grew with videogames).

In other words, when I am highly focused in these activities, doing something I like (oftentimes) and I strive to do it the right way. Usually I enter Contributor Mode to have a break between 2 different job activities, to relax a bit.

This makes distinguishing among work and contribution periods easier. Often these two activities may be on the same level of importance, but I prefer shifting my attitude in performing them, for later I may use the outcome of contribution for a job project.

Being able to split way of thinking it’s very important in understanding the differences, the quality of what you are doing, why you are doing it and also it helps appreciating more what you do.

The contributor, basically, never forgets why he is doing something: A contributor needs a goal to point to as a North Star to be **motivated** to do things, not just for the sake of them, but because of the desired outcome.
To make the goal - that for a contributor is not earning his lunch - more enticing and motivating, there are different practices to improve the quality of the time spent. Don’t forget a contributor is using one of the most valuable currencies (not speaking about blockchains here) *time*.
This currency needs to be spent wisely, the boring parts of the workflow, or the ones prone to express more issues, need optimizations, it’s better to prevent issues where more attention is needed and also an understanding of how to delegate or ask for help is needed.
Question is how you can move in the Open Source world in a comfortable way, starting by doing something that not only helps the project, but is also enjoyable, interesting and thrilling without abandoning the floor?

Simple! You have to learn to live inside it.

How can you live in an open project without any previous experience in contributing to one? Consider that today open source is not anymore just for hobbyist/hobbits but for professionals, check job offers in the IT if you don’t believe me!

Anyway it is a good question, if the answer was simple you probably would be doing something else now, instead of reading this book. Also don’t forget that the smaller the project is the easier it is to contribute to but you should not be afraid to discover how big project works.
Take a moment, relax, get ready to discover a new approach to community projects that may help you in your work as well as your daily life.

Donating money is not enough

Money is never enough, but in open source this is not a real rule. It is quite common (and rumor) that many projects have so much money that they have no idea on how to use it or projects that doesn't need them because there is a huge community moving on, and they need to cover only hosting expenses.
Open Source doesn't need only money and promotion, but contribution to itself. Usually the enthusiasts contribute with these because it is more simple, but how you can feel part of something without doing more?
Anyway there are cases and cases so if you want to contribute with money in FOSS there are various ways:

- Hire committers: As company you hire or pay someone to dedicate hours or all their time to work in a project, not so much different on "normal" working
- Bug Bounty: They works rarely to require a different approach like you are an employee with receipts but also reports that will take a lot of time for someone that works on his free time (an old GIMP story[81])
- Donation: Just to be sure that is the official way as often it is possible to donate to specific contributors on their Patreon/etc[82] instead to the whole project, so just think what is the best move, like I need an invoice, or I need that change
- Certifications/Books: sometimes it is possible to get official certifications for employee or books by the project's authors
- Pay for changes: similar to the donation but it is possible to take a consultancy company that will work on the project and to the patch, with moving on the integration itself
- Associate: often behind a project there is a foundation or an association that manage the money and the future, you can associate and be part of their decisions

A fact very common it is company contribution improve the brand of the company itself and also that today the employee are more picky in the job opportunities. They are like to get the best job that let them to do what they like as example 20% of their time to contribute or work in their projects.
There are many projects started as private or side project inside a company or during the company hours that became famous and provided a way to find new hires or the company status in that sector.
Or looking for free labor in their own projects, but without reviews is useless.

Let's see an example of OSS project that forced the entire ecosystem with Chromium[83]:

- Apple forks KHTML (from Kounqueror by KDE) and creates WebKit
- Google creates a new browser based on WebKit (and hires various Mozilla employees)
- Years later as Apple is not following enough all the changes by Google they fork it and create Blink

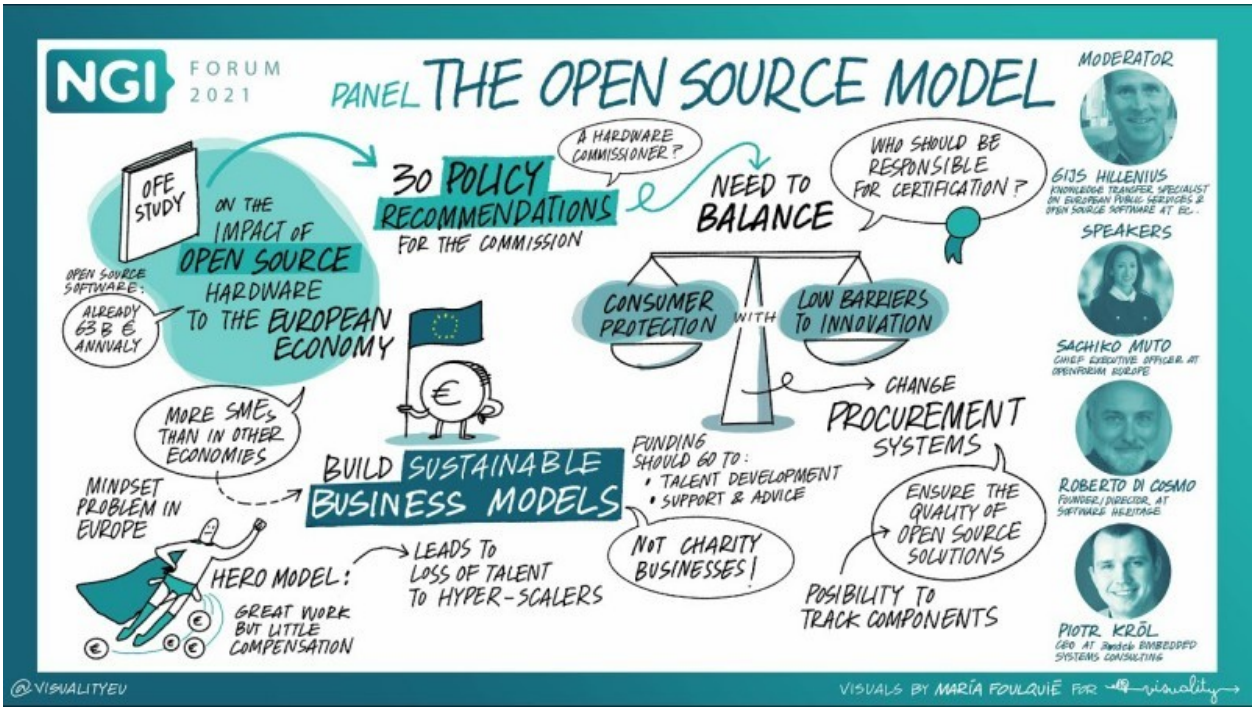
[81] http://dneary.free.fr/gimp_bounties.html
[82] https://www.youtube.com/watch?v=MzpoTuHuciE
[83] https://www.reddit.com/r/opensource/comments/phm308/comment/hbjo95j/?utm_source=reddit&utm_medium=web2x&context=3

- All the browsers (except Firefox) as today are based on WebKit or Blink (often is branded as WebKit)

Now for them is easier to push standards or bad behaviour as their is the most used technology in a ratio 20:1, so now with the new Manifest V3 for extensions[84] will be more difficult to do adblockers.

This is a bad way to use an OSS project but is also the reality as there are many good examples that is not the case to discuss or this will a new chapter.

The HR world is looking a lot in Open Source to find the best person for a specific role, as everything is transparent is easy to see what are the real skills on the field that can be coding or like localization (a company don't need a localizer for any language but a manager that know that field as example that can open new market).

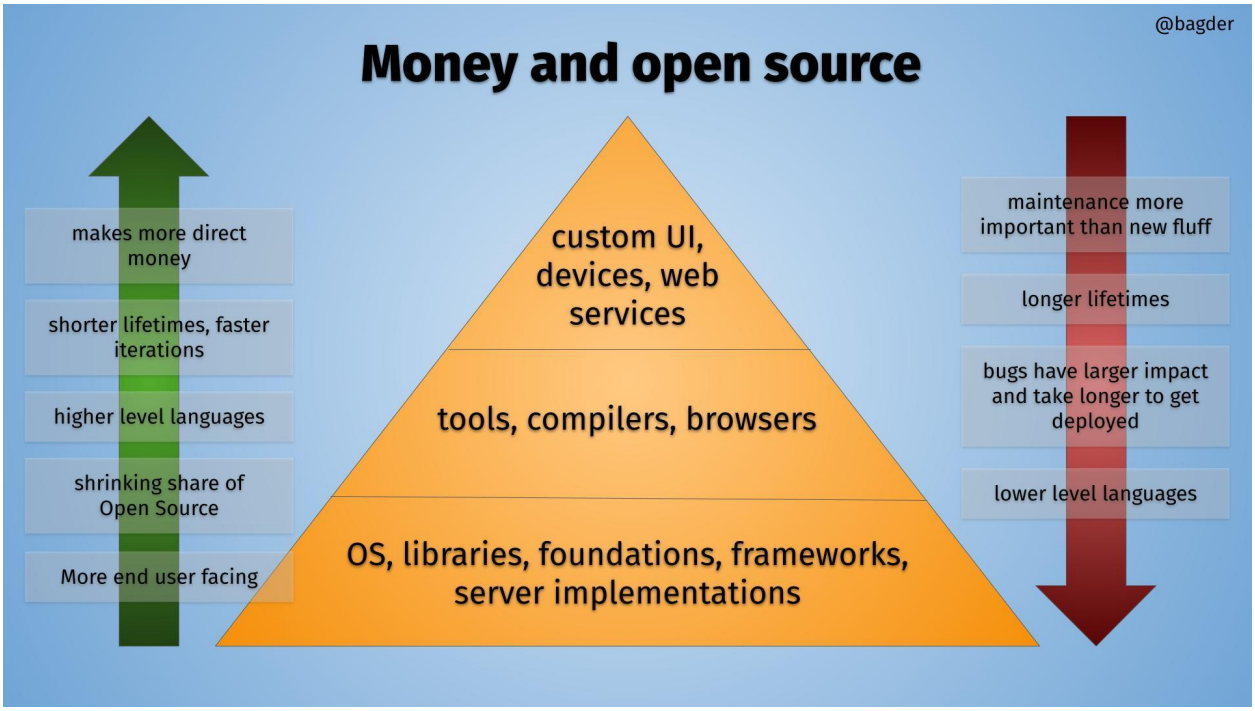


The Open Source model by Europe Union

From a point of view of the Europe Union we saw that there are other values we have to consider just the money and the impact that they have in the whole sector, like new opportunities.

In conclusion if you are looking on other ways a project can get donation or money take this KDE example[85], considering that has community they have funding for events, but contributors have their Patreon as example. If you are looking for ideas about jobs roles in companies about FOSS, take a look on this report[86], they are looking more on developers than community managers.

[84] https://blog.mozilla.org/addons/2022/05/18/manifest-v3-in-firefox-recap-next-steps/
[85] https://quickfix.es/2021/08/the-three-laws-of-floss-projects/
[86] https://www.trueup.io/open-source/reports



Daniel Stenberg's "the OSS pyramid"

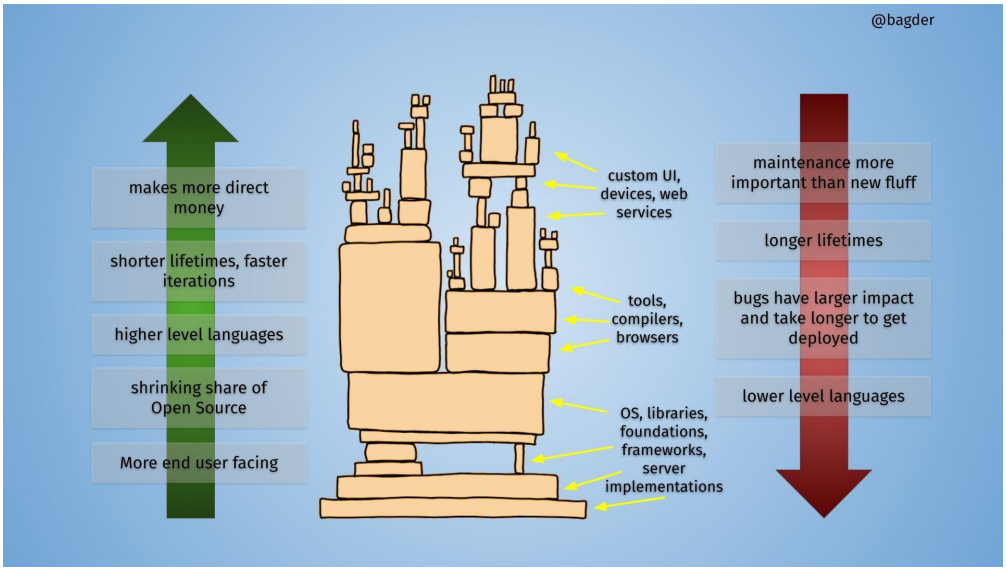
Daniel is the author and maintainer of cURL (started in 1996), that is the second OSS project most widespread after Linux. Think, Linux is on TV, refrigerators, routers, servers, smartphones, spaceship and so on. The second one is a library for all the operative system (with a cli tool) to download and upload anything on every internet protocol that is used as base for other languages or other tools. He has a lot of experience on the project and on this topic, he worked before for Mozilla where his job was just to work on cURL (and isn't used inside Firefox), and he introduced this "pyramid"[87], the idea started on the security topic.

> Inside the pyramid there is a hierarchy where things using software are build on top of others, in layers. The higher up you go, the more you stand on the shoulders of open source components below you. At the very bottom of the pyramid are the foundational components. Operating systems and libraries. The stuff virtually everything runs or depends upon. The components you really don't want to have serious security vulnerabilities.
>
> In the left green arrow, I describe the trend if you look at software when climbing upwards the pyramid.
>
> Makes more direct money Shorter lifetimes, faster iterations Higher level languages Shrinking share of Open Source More end user facing
>
> At the top, there are a lot of things that are not Open Source. Proprietary shiny fronts with Open Source machines in the basement.

[87] https://daniel.haxx.se/blog/2022/01/17/enforcing-the-pyramid-of-open-source/

> In the red arrow on the right, I describe the trend if you look at software when going downwards in the pyramid.
>
> Maintenance is more important than new fluff Longer lifetimes Bugs have larger impact, fixes take longer to get deployed Lower level languages
>
> At the bottom, almost everything is Open Source. Each component in the bottom has countless users depending on them.
>
> It is in the bottom of the pyramid each serious bug has a risk of impacting the world in really vast and earth-shattering ways. That is where tightening things up may have the most positive outcomes.

The idea is that the most bottom stuff require more attention by moneymakers, like companies, as they are dependencies that can affect more scenarios from security to maintenance (like bug fixing or new features).



XKCD (2347) "the OSS pyramid"

This version maybe is easier to understand how a component in a business can be valuable but in Open Source is ignored by those entities as they are not contributing to the future of this tiny piece. Or another short version is Sponsoring dependencies: The next step in open source sustainability[88], that makes more sense, after all every OSS project is based on other OSS projects that often don't get the credit deserved.

How to live inside the Open Source

First step is discovering the **Code of Conduct**, usually every project has one and they are based on the same basic rules, mainly covering how to work on the project and relate to others among the

[88] https://humanwhocodes.com/blog/2022/06/sponsoring-dependencies-open-source-sustainability/

project.
In detail, common sense rules: not to be aggressive, no bad words or insults, understand contributors come from different backgrounds and it isn't always easy to understand each others or avoiding conflicts without applying patience.

Being Italian, for me it's normal to make gestures, even when people aren't seeing me. Other people may be confused by that, or the gestures themselves can mean something different for them. Think about that in a big picture, adapting your tone to the environment.

This Code of Conduct often applies only to the project home and not for interactions with others about the project happening outside the community (sometimes you even have to sign the CoC), so a twitter interaction may violate it!

For Italian people, I usually summarize like follows:

- No bad words
- Don't insult other people (except the ones eating pineapple pizza or drinking cappuccino during a meal)
- No gestures

For Italian people usually those rules are enough to understand how to behave differently, maybe the rule about gestures is drastic but it's a very important one (of course depending on context).

Body languages difference all around the world

Basically Open Source is a group work with different people so is important to have some basics rules to live and moving on and avoid problems during the discussions that can degenerate on flame war. A reading about work with people that you don't like can be easily found here[89].

Understand if a project is alive

One of the common issues in Open source is to understand if a project is alive, abandoned or dead.

When is alive is easy to see, the tickets are getting replies in reasonable time (like 1 month at maximum), the pull requests are getting reviews, the social channels keep pushing updates when there are (and not forgetting to do so) and so on.

Abandoned is more complicated because maybe the owner or maintainer is not replying on the tickets or pull requests but is active on social networks. So basically seems that is not caring at all of the project but he wrote anything about it in the project page itself. Usually this means that a fork is possible without so many troubles but probably will be difficult to get the ownerships of the website domain, social accounts etc from the original creator.
There is also the issue of the naming issues of a fork or the trademark, we know how much is important the naming of a project today.

"In software engineering, a project fork happens when developers take a copy of source code from one software package and start independent development on it, creating a distinct and separate piece of software. The term often implies not merely a development branch, but also a split in the developer community, a form of schism." Wikipedia[90]

Instead dead is like abandoned but there are no forks or any communication from the authors but just people asking if the project is still alive and no one is interested to move on the project.

Based on your skills you can decide what kind of effort you can put in a project and define a goal. As example can be: create a new fork that handle some bugs that are very annoying (to still use the project without do any brand changes) or define a big plan maybe with the help of someone else.

As newcomer to open source of course probably is better to start with a project alive but if you have skills and you think that you know how to lead a project, keeping the quality standards and so on with a fork.
The kind of fork is based on the goals that you want to achieve and the time that you want to put on that. Can be strange this reference to time but when you take ownership of a FOSS project you will see a lot of problems with time management for various reasons but this book has a full chapter about that.

[89] https://hbr.org/2018/12/how-to-collaborate-with-people-you-dont-like
[90] https://en.wikipedia.org/wiki/Fork_(software_development)

How you can help the Mantainer

In case it wasn't clear, this role is the one that covers the project's updates, communication like replying to tickets/pull requests, doing code and so on. Basically it is the Janitor but also the Artisan and the PR of the project in a single person. It is a busy role and depends on and how the project is structured because this role ca be very time-consuming and generate easily burn-out.
Too many tasks for a single person that involve different areas and is difficult to find a person who is good in all of them, maybe is a person that speak a lot of jargon or just check the project's notification once a month or just add the code from others under his name, like in Vim[91].
Maintainers need to scale with the project growing, and this depends not on the project's users growing but on contribution and interaction on the project itself. There are maintainers that push every week something in their projects to show that they are alive, or others that with help from contributors/employee share duties.
Another non-written golden rule that I remember from Mozilla is that "contributors do stuff for fun, but annoying things are for employees", this means that in Open Source you are talking with someone who is doing on his free time (the majority of times).
How you can check if this Maintainer is working with a salary on the project or not? Check is website or social networks, so you can understand with who are you talking.

A story from the creator of ESlint[92] about the duty of a Maintainer[93] as bullet point:

- Girlfriend asks to buy a puppet together
- They are not an official couple
- The puppy will leave with him
- The duty of taking care is to him
- Her will just take the funny part when together (like walking)

So the duties are on only on the Maintainer and the others will use and asks to keep the project moving on, with new feature requests or bug fixing without any real help. For is looking to a good end, no the puppy wasn't brought to home for the duty reasons non-shared.

An example of burning out maintainers is being on track with the dependency of the project, as they can break it as it is not possible to manage all of them. We are talking on the duties of Maintainers and there was a comparison with the financial crisis[94] that basically says that there are many variables that create entropy that as leader you can't handle, but you need to manage.

We know that who will check our contribution is a normal person with different time amount available to look on it, so asks yourself:

- How this can help the project
- How can I help to push this change in production/next release

[91] https://github.com/vim/vim/graphs/contributors
[92] https://eslint.org/
[93] https://twitter.com/slicknet/status/1430334633611202562
[94] https://blog.scottlogic.com/2020/12/22/software-crisis.html

The first one is easy as can be a bug report, a localization, replying to a support request etc but the second one is the real question. As example:

- Bug report: as we already said giving all the information to replicate the issue, an example user case or why I got that error and what I was doing
- Localization: check all the translation guidelines and verify that you are using the same word used in the other localization for your language
- Support/Promotion: in case a request is happening, often push that to the Maintainers to do something about it or create some documentation for it
- Coding: propose how the code change with lines numbers or do the patch with all the linting/etc done including explanation how the changes works

When you are contributing is important to you to act as a friend and not like a customer to the Maintainer so provide all the help in what you are asking, in this way you can grow your role in the project but also your knowledge/awareness[95]. Remember more are you autonomous and providing a "complete" task in a project more you can drive it in the direction you want, that can be a feature that you need or finding new contributors.
The first expectation in contributing is a *hope* that your contribution be handled, and you need to help this hope to succeed, like for our daily hope.

Communicate on the internet

The action of communicating is fundamental. If we communicate badly we loose everything. The talker is an introvert who may as well work on his own, all alone in a garage, without issues.
One reason why I wrote that book is my interest in life coaching and self-help, that helped me discover many things about the fact I am not so good in communicating what is on my mind and to meet what others expect in terms of interactions, tone and mood from me.

Think about being in an international context, where there are cultural differences, and communication happens in another language to understand how complexity level can ramp up.
We will cover various kinds of communication in Open Source, but before that, let's refresh the basics:

- **Be proactive**: while contributing to a project, rants are useless (and may hinder you from reaching your goal). Keep every request open, friendly and offer options.
- **Transparency**: doesn't mean that everything should be participatory but just results published (maybe with a private invites)[96].
- **Don't be passive**: don't avoid discussions, don't ignore people. This often happens for shortage of time, but may damage you in the long run.

[95] https://notes.eatonphil.com/learning-a-new-codebase-hacking-nginx.html
[96] https://blog.opencollective.com/php-foundation-alive-and-kicking/

- **Clarification**: Not all readers may have your same background/skill set or knowledge about the project. Those people will produce noise and confusion in trying to understand more (even if they are perfectly able to google it), write complete answers to them (the trick here is including external resources to compliment your answer)
- **Documentative**: every message, if carefully written, may turn out to be a resource to be reused in the future or an historical archive. Strive for completeness in writing.

Few example of this points on public discussion I started:

- About gratification for volunteers in Mozilla[97]
- About regional local community management in Mozilla[98]
- About the Volunteer role in the Mozilla workflow decision chain[99]

The lesson of this section is: don't be afraid to ask questions (in the right place).
Open source contributors also have a big problem about communications because you need to be also a good communicator. **We are exposed on the internet**, what we say as comment about a project or a issue can be the first step of the Butterfly effect[100]. We prefer async communication, especially if we are not native English speakers because we can improve the quality of what we have to say and be more comfortable.

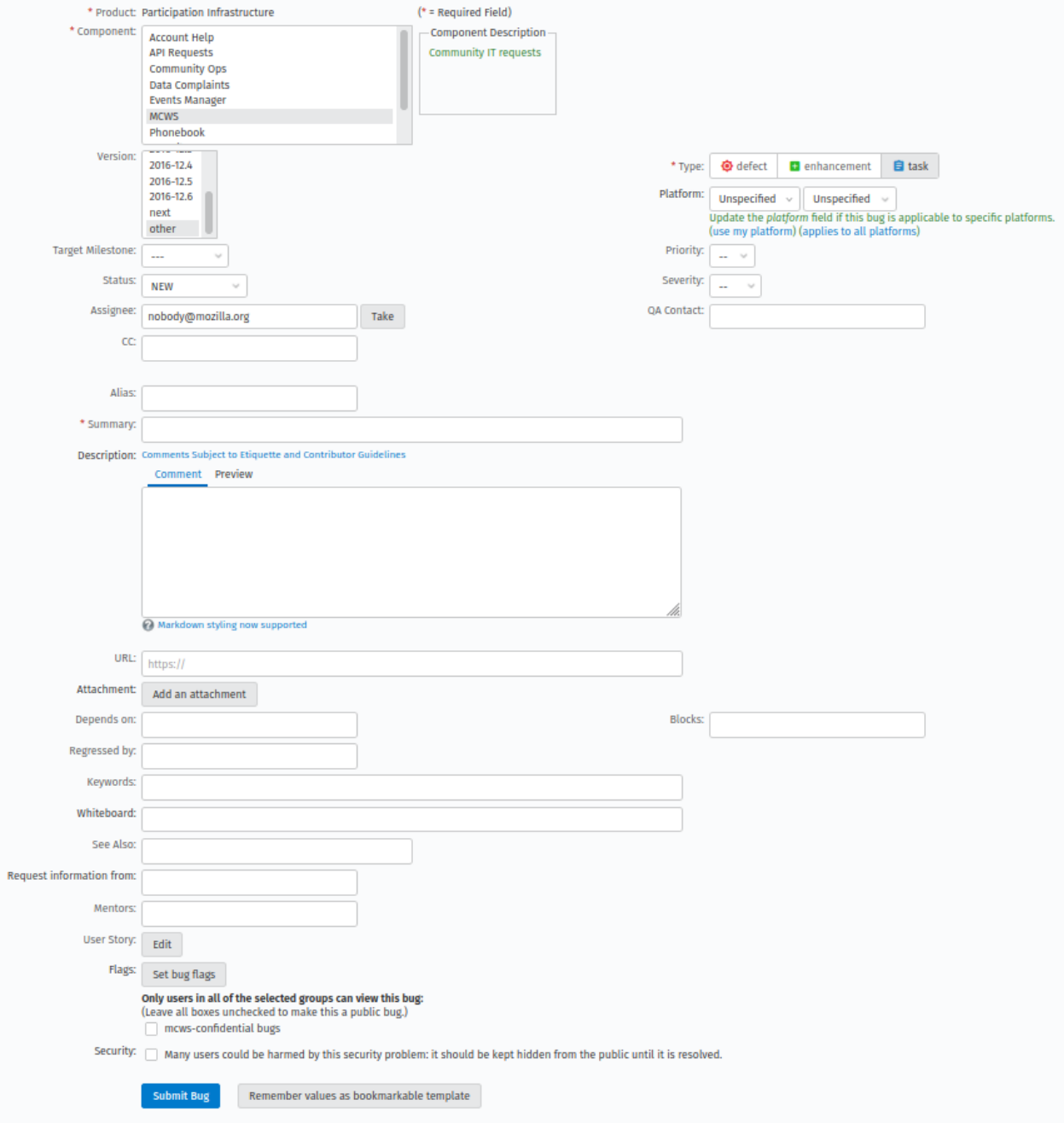
An example of don't be afraid is John Carmack[101], one of the authors of videogames like Wolfenstein and Doom and pioneers of releasing as open source famous video games. Today he is the Consulting CTO Oculus VR and a famous developer from all the innovation in the 3D world and the promoter of Open Source.
Anyway as every single person in front of something is not clear, he started a discussion that got popularity on HackerNews[102] discussing how also a person like him doesn't understand a C++ code and just like everyone wrote an email to the mailing list of OpenBSD.

It's perfectly fine to admit ignorance in something, especially when you are starting.

Ticket/Bug/Issue

Based on the service/tool used they have different names, I prefer *ticket* that can be a container also for other kind of requests like features, roadmap, discussions instead of something specific (like a bug).
Opening a ticket results in 2 things: advanced knowledge of the project and no fear of about the tool itself.
Writing tickets for a lot of people is the most difficult thing, so I want to start with this one.

[97] https://discourse.mozilla.org/t/about-gratification-for-volunteers-in-mozilla/38181
[98] https://discourse.mozilla.org/t/about-regional-local-community-management-in-mozilla/38924
[99] https://discourse.mozilla.org/t/about-the-volunteer-role-in-the-mozilla-workflow-decision-chain/38451
[100] https://en.wikipedia.org/wiki/Butterfly_effect
[101] https://en.wikipedia.org/wiki/John_Carmack
[102] https://news.ycombinator.com/item?id=23224584



Mozilla Bugzilla

What does advanced knowledge of the project means?

- Terminology
- Knowledge of another language
- References to classify the request itself (set the various fields about the technical stuff)
- Verify if it is a duplicate
- Give suggestions too

- How to replicate the issue
- Confirm the issue (replicate it on your own)
- In case for performance of the action by specific people

First of all recognize if the request is compliant to the reporting tool. Few examples: ticket about an issue that instead should be reported to another project, documentation issue reported as error instead it's a localization issue, budget request in a public ticket system and so on.

Scared about the tool?
Often the tool is public, sometimes there is the fear of the public (like in public speaking) with concern it would be permanent because it is written. I remember the first time I opened a ticket on Debian for an error on a software that I was using. I had a lot of anxiety because I had to use the email instead of a web interface, I was writing a ticket to one of the biggest Linux organizations in the world.
I was thinking along the lines of: "it's a stupid error", "maybe it's my fault because I don't know how to read documentation", "I am reporting to the wrong people, since it's the tool and not the distro", "maybe my report is lacking something important", or "it's poorly written and they will mock me for that".

Basically I was joining the public circle of technicians that contribute to one of the most famous project in Linux, it was public and open but at the same time it was for a small elite. I took a breath, provided more information I could, verified it few times and sent it (without asking for help, all by myself).
Cannot recall what the ticket was about now, this isn't the point however; after that situation I had no more fear or anxiety. If I, an expert and skilled person, was to have them, it was easy to understand what newcomers feel in front of a form whatsoever (even when posing the simplest questions).

Another point is finally find a ticket about your issues/feature request and reply asking of updates (maybe is very old), as we will see later in the book with a dedicate chapter, usually maintainers or other people doesn't have time to keep up everything.

On the other hand who will read the ticket need to be a bit of a community manager not just a project maintainer. Closing a ticket because there is no time to do it is wrong, is evil for your users and a misuse of the tool itself.
Who opened a ticket, as we saw, require for newcomers a lot of trust on their skills, so your reply is the first touch with someone more skilled than them and what you will do will impose the "behavior brand" of your project to them. They can enthusiastic people that want to help or someone that cannot use the software maybe for configuration issues or a bug (they always think it is a bug).
One of the common comments that I receive on tickets in my project as example is a thank you because I replied (this is not so common in OSS) and also that I replied very fast (in the same day at least if now few minutes). People approached in this way feels more appreciated and can be very helpful and motivated to do more, is a win-win for both parties.

A key point to involve new volunteers if there are information about how to fix it, like a step by step about "what is missing to approve and close" so the ticket will not be abandoned for months because the status isn't clear. And again there will be a documentation of what is required to do.

On the other side if someone open a ticket give to them the clues to fix on their own the fix and contribute back to the project. One example is giving them the line and the file to patch so they can open a pull requests or suggest a new wording for the documentation.

Email

Mailing lists are the noisier communication medium, it's easy to lose interest or feel in the wrong place.
It's very easy to find people using *Reply to All* feature just to ask what an acronym means or what kind of tool are you referring to and the like. Netiquette is so easy that often… nobody uses it.
Let's imagine a "Thank You" being sent to all the thousands participants in the mailing list, for an update or piece of news that was shared, it's better to send that kind of messages in a private way.
My colleague Eugenio often says he wants to write an one page book: "*how to use the reply all feature of your mail client*". Often if the conversation is among more than two people, they forget to use it failing to spread the message across, or, on the contrary, sometimes they don't understand how much noise they are creating mailing to everyone something that is meant to be shared in another place.

Different issues, same communication medium with different rules for each kind of communication that has to be shared. That requires a bit of thinking along the lines of: "am I writing this in the right place and with the correct format?". This may save you time, protect your pride and also your skills won't go wasted, for one of the issues people face with this technology often is: "why do I have to join these discussions?"

Mailing lists are "topic oriented", and for some people that is enough. But you need to understand who actually is going to read these messages, readers may be developers, localizers or just fan of the project, You need to give exhaustive answers, as said above, with links to complete what you say, and an invite to reach out in private in case the correspondent may have further questions.

Next question is: how can we be sure that mailing list doesn't get ignored/rarely used and that when it is used it is functional?

Let's enter copywriting realm, few rules just for you:

- Use the 5 W (who, where, what, why, when), and in addition to them, the intruder "how"
- Split the content in topic based sections
- Put the request for action at the end as a bullet list (easy to remember, gathers interest)
- If the message is long, include a short recap at the end of it
- Better to supply reference to previous stuff/explanations as links not to crowd the communication
- A remainder about Netiquette may be advisable on some kind of posts for updates

Visit these links to find examples:

- Bits from Debian (February 2019)[103]
- Bits from Debian (January 2019)[104]
- WireGuard: Secure Network Tunnel integration on Linux Kernel (advanced)[105]
- Counterargument to Deprecate Short Tags RFC V2 in PHP[106]

Just as note Linus Torvalds was interviewed about how he managed the Linux project from home with mailing lists[107]. Basically be short but check few times (before to send it) what you saying and be clear.

Another point is that they are old, new projects that have less of 10 years doesn't have one so from a side there are communities that migrated to Discourse or people that still uses with custom mail server configured to organize what they receive with those. It is something that you can see with veterans, but is not something that is scalable and approachable to newcomers.

An analysis of this technology with the GitHub (for coders)[108] shows also other points of view, like:

- Discuss a patch on mailing lists requires diff that often doesn't have syntax highlighting and various threads as the patch get updated, so the final version in the various email exchanges can be everywhere. Instead, a centralized solution lets an interactive code review with a history that is more simple to navigate compared to various patch sections splitted in various email in whatsoever order.
- GitHub offers an API so in case you can do your own client or a different interface based on your needs compared to emails.
- As it is a centralized solution you can't be sure that a copy of that code will be forever in that place (usually big projects have mirrors for this reasons) but this can happen also by email as it is to impersonate someone else
- Pull requests doesn't allow various authors on the same one (unless you give to them permission), instead by emails this isn't a problem but means more work for the committer to gather all the wanted changes

To these points I want to add that on GitHub/GitLab/etc you can disable notifications by issues/pull requests etc instead by mailing lists is not possible, so understand what to read can be very time consuming. Honestly, this is the reason, why i am subscribed to very few mailing lists (not just about coding) and enable the daily or weekly summary.

Chat

These days, they took the place of mailing lists condemning them to oblivion. Especially the new generations find very uncomfortable to write long messages, considering the fact they very often

[103] https://lists.debian.org/debian-devel-announce/2019/02/msg00010.html
[104] https://lists.debian.org/debian-devel-announce/2019/01/msg00010.html
[105] https://lkml.org/lkml/2019/3/22/95
[106] https://wiki.php.net/rfc/counterargument/deprecate_php_short_tags
[107] https://www.zdnet.com/article/pet-the-cat-own-the-bathrobe-linus-torvalds-on-working-from-home/
[108] https://asylum.madhouse-project.org/blog/2018/07/24/on-git-github-and-email/

use a smartphone in place of a computer.
Sometimes however they behave silly, for example when they ask what a link is about without even opening it. There is no way around that, the only thing we can do is welcome them in the community as fast as we can, so that they may learn its rules.
Rules may be boring to write down, but are indeed important. We need to explain them and enforce them as mandatory to avoid spam and people abandoning the channel for it's impossible to catch up with it.
It may be scary to some people to receive many notifications of unread messages from a very crowded community channel.
It is important to manage the volume of communications in the channel, remind people to stay on topic, be relevant and respect the space and time of others. That is why messages need to be clear, because it's easy to ask clarification by e-mail, but imagine on a crowded chat where going off-topic is a real risk and getting back on track after a "wreck" of non relevant noise may be a nightmare.

The big issue with chats is to stay on topic, and get back to the topic when the situation gets noisy. It's easy to change topic with a single message, or split a discussion in many little rivers among a group of people. **CHAOS!**

An example of this is when one piles many questions/issues in one single message, when people join after some time the discussion,they are unable to catch up from the very top sometimes. In that case is better to face one thing at the time and pass on to the next when the previous is exhausted and in doing that there is the need to be intransigent with the discussion remaining on topic.

E-mails allow for recap, or a single notification for a group of mails, chat systems send a notification for every and each message that gets to be posted, this resulting in a dancing and ringing smartphone in less than a minute.

For this reasons a chat solution is fine for some stuff and not for everything. Also understand the kind of technology for chatting is important, is not just a flow of messages like IRC (that anyway is getting abandoned for various reasons), but include usability, multi-device usage and various levels of anonymity.
When asking people to use IRC or Jabber, they are confused to have to configure a "client" based on their system. In today's world, people use apps which do those things for them. Plus, these clients don't have modern features like the link preview with an image, title of the pages etc, bot with buttons to press instead of commands etc.
Another problem is anonymity of today, since the various Code of Conducts require actions to be enforced against persons/users.

There was a nice talk about the next generation of contributors that don't use IRC (and we have to accept it) at FOSDEM 2020[109] .

So a chat is not just a place to discuss but part of the ecosystem and need to follow some rules and expectations that are shared with everyone. In fact as example if you take specific kind of community for they is perfectly fine to be anonymous and they don't care of the client etc but for others instead the appearance is very important.

[109] https://fosdem.org/2020/schedule/event/nextgencontributors/

Public Speaking

Conferences and events are a fundamental part of the Open Source world, because they allow networking among people, people have fun and meet people in person (after knowing each other online for years). To talk in public is important for it can be easily recorded, but if opening a ticket may scare a newcomer, go figure veterans also may be scared from public speaking.

Years ago I wrote a guide on Public Speaking (now hosted on GitHub) and I am attaching it to this book as an Appendix.
I don't want to cover how to better speak publicly, but how to win the fear of doing it, because it's one of the common issues new volunteers face, especially at their first ever speech.
At your first speech, you aren't speaking in front of the president of your country, but you are in a niche environment - surrounded by other friends, meeting with like-minded enthusiasts or in a not so crowded venue. It is the ideal kind of setting, because it helps keeping the stress to a minimum and boosts confidence.

Another way to boost confidence is to try the talks in front of your computer, like when, as a child, you had to learn poems by repeating them on and on. You practice and you get better at it, with the bonus of not having a teacher or parents watching your performance, but friendly faces wanting to listen to your talk about the topic without being interested in you.

Only growing up as a speaker, people will have expectations on you as a speaker and on the topic itself. Right now you don't need to bother about them having expectation on you, however. In time you will win over this whole expectation thing, but now you don't need to be scared about that.
It's like learning to ride a bike, it's done step by step. Start by selecting a topic you are interested in and know pretty well, try it, ask for others to revise it and try it again.

Like I said in my long biography, for me the turning point was speaking at a Linux Day in Italy many years ago. I fought many fears, even the one of joining the community in person. From that moment on everything was a downhill, with brakes on of course, because one needs to prepare for things for the Open Source word despised unprepared things, its philosophy isn't allowing for that to happen. We are forced to know what we are doing because others will see that. And that is a little thing, that often doesn't happen in private companies. The absence of that principle often leads to a toxic environment.

Document it (even if time consuming or boring)

One of the issues one can face on the internet is when looking up the solution to a problem, one may find only very old content on forums, with a queue of people confirming they had the same issue over time under that discussion, but no answer about a solution at all. Usually one of last messages of that kind of discussions states "problem was fixed" without any real documentation of the solution itself, so that you are left screaming at the moon in frustration.

XKCD #979

The Feynman technique[110], created by the Nobel winner physicist consists of 4 rules:

- Pick a topic you want to learn
- Pretend that you are teaching to a kid (like the hashtag #explainlikeiamfive)
- Identify issues in your explanation, go back and recheck
- Simplify and use analogies

This technique is used to learn any topic in the most effective, easy and fast way. There is another version called Explanation effect[111] that include also a research and other modern information.
In Open Source world those technique would be a part of Public Speaking of Documenting.
Being a grumpy cat doesn't clearly help, even if you are trying to use the technique. So it is required of you to come out of the shell, meet people and confront with them. Trust me, at the start it may seem it makes no sense, but one of the reason this book was started, was because of that.

What are the benefits of applying the art of documenting then?:

- You save time in the long run
- You improve communication skills
- You face different people with different backgrounds and skills
- You help improving the project's branding
- You provide useful resources for newcomers
- You feel confident for you know the topic or you have a real experience with it
- You will getting a growth mindset (about this topic there are various books)

Growth mindset is the opposite of the Fixed mindset that is based on the assumption that skills and knowledge are predetermined. This mindset enables you to love to learn, challenge yourself and experiment, so the performance and personal skills improve.

[110] https://curiosity.com/topics/learn-anything-in-four-steps-with-the-feynman-technique-curiosity/
[111] https://medium.com/accelerated-intelligence/explanation-effect-why-you-should-always-teach-what-you-learn-9800983a0ea1

Also to keep you active and aligned with your world is important to not stop from learning every year something new, and maybe learn how to document can be your next goal to move on your knowledge! Don't forget that if you stop to learning you can become easily obsolete but the goal of this book is to not be so pessimist.

Those are the good reasons to start documenting the right way, let's see them in detail!

Lets you to save time on the long run

Common situation, an user not very active in the community posts a question on the forum linking to a previous discussion including all details of the issue he is asking about. Ignoring the quality of discussion, that is a clear example of time saving.
Picture this in a huge environment, where all the media we saw above are used to interact, documenting something allows volunteers to avoid asking the same stuff again and again, avoiding the consequent demotivation process. It's easy to get mad when using our time to constantly tell people to shut down and restart.

Did you tried turning off and on?

That's why FAQ or manuals are written, to gather this common questions and tasks to save your time.
Don't forget that forums are disappearing online because search engine let you find these manuals or guides without losing time on waiting somewhere and asking for a reply. So writing stuff is very helpful also to be more easily found on the internet.

Improves your communication skills

True story, this is never going to end in the transparent Open Source ecosystem we live in.
Knowing how to write useful stuff for others requires knowing the issue's ins and outs and at the same time being able to express every detail, and also the very common "check the version of the software you are using".
Being objective, avoiding off-topics, being essential and understanding which are the requirements of the project do help a lot. If you don't understand the project needs documenting, there you aren't listening to your users, after all. There are people looking how to shut down their computer on the internet…

You don't need to be J. R. R. Tolkien (polyglot, philologyst, writer), Ernesto Bignami (he lended his name to tiny books that recap different kind of topics like maths or story), Isaac Asimov (high IQ, famous writer of books in different topics), Bruno Migliorini (linguist, phylologist, Esperanto speaker and author of the first edition about the story of the Italian language, president of "Accademia della Crusca") or Dante Alighieri (the author of Divina Commedia).
You can always improve your communication skills, basing on the context, by looking what others are doing differently and learning from them.
And even if you aren't able to see differences, this doesn't mean there aren't.

Lets you meet new people with different skills and backgrounds

We can say the previous point allows you to understand the various kind of users in your project community. From a marketing perspective, if you are working on a project aimed at hospitals your audience won't be composed by farmers or plumbers, but from nurses and doctors. This means you have to adapt your communication tone from terminology, to the way of explaining IT stuff, to details and understanding when they aren't needed like when explaining acronyms basing on the context you are in, and you also need to have basic language skills.

In marketing that is accomplished by creating the various Personas representing ideal users, you can use these to shape the flow of your project and the communication tone. Something we can learn from the Startuppers, they need to adapt their product/service to an undefined audience, understanding what customers need is the first step in shaping their Personas and is useful to shape the best product/service for them.
In short, levelling down the learning curve is very important. Documentation has all this power, and gets unused. Let's move to the next item…

Let's you improve project's branding

Usually project become famous for they are the first to appear on the scene (or the easier to get access to, like Internet Explorer) or because they are most friendly. Not for their quality in both cases.
Make sure to include resources to make the project more intuitive also, if project itself is complex. A simple example of this is Arduino. We are talking about electronics here, as of today Arduino has many competitors, but Arduino stand out among them (and is default technology supported almost

everywhere) because internet is packed full of information about it, also libraries to implement it can be found over the net and many events are organized by third parties to spread the world and inform/educate people about it.
Another example is ArchLinux in the Linux-sphere. It's so famous because it includes an extensive documentation that may also be used in the context of other linux distros for its completeness, among other things.

In short, documentation helps to sell, also in other contexts. If you want to address specific niches it is a requirement, for you can't convince people by posting stickers and cool pictures over the social medias, people want facts. Documentation is a fact. Its quality shows the completeness and coolness of the project.

Very often you aren't in the position of the first on the market or the more available, but you may be the more intuitive. Again, documentation helps a lot because it allows people to do things on their own. That is the goal the end users want to achieve with the technologies they use. One of the reason why Open Source project grows and spreads wide is they are autonomous, not depending on others support and being able to rely sometimes on periodic fees charged for their usage.

Provide useful resources for newcomers

Newcomers always have the same issue as we had ourselves in the past: they are disoriented. Transforming them in autonomous users is critical to lower their defenses, it's the reason Travellers' Club is famous, it lets travellers collect books about the cities they visited, to visit them as they wish. In an Open Source project is not always possible to do what one wants, having a set of instructions is very helpful. It's like the concierge in a hotel explaining what to visit, where to eat and so on. It's nice and one feels welcome.

One of the suggestions or activities I think are among the most powerful to boost engagement in newcomers is proposing them to read the documentation because:

- They discover more about the project, they get trained without losing time on mentoring/supporting them
- They review the resources and may be able to signal if they need updating/corrections, if something is missing and if they are overall clear and understandable
- Tomorrow they will know how they can use documentation and they will help in empowering it in the long run

Reading is accessible to everyone and doesn't require a lot of efforts or time (not to mention by reading one gains experience on the project) but it's a first important step to engage and involve the reader in being a part of it and to contribute back. Maybe fixing typos or updating screenshots included in the page.
Read the documentation let the volunteers to discover how the project/community is structured, organized and what are the various web portal or tools that are important. As example in Mozilla, Pontoon is the web tool for localization instead in WordPress is GlotPress or in OpenSuse is Weblate.

It's comfortable because it's something you know about or have real experience on

Homework at school could be difficult for someone (in Italy as students we also undertake oral interviews one-on-one with teachers) and may require special studies that after short time are easily forgotten.
Instead, writing what we know from our first hand experience is easier, because we already did it, we know the difficult parts and we can help others and at the same time improve what we know.

It's not required from you writing documentation for something you don't know about, so you can't ask newcomers to write documentation themselves as their first task. This is something for veterans, like you are now (or will be in future!).

Encyclopedias are written by experts, and we are the experts of the project. Sure, Wikipedia makes this statement feel outdated, but consider a bigger ecosystem where you are part of the big picture.

Don't stop to learn

Don't stop to learn in a world that moves quickly is important, to keep your brain elastic and fresh but don't lose the motivation to improve yourself every day. We saw that the first step is to teach to others but there also other steps or tasks that you can do, inspired also by 22 Simple Ways to Learn Faster[112]:

- Read a lot of articles (every day or every few days): use Reddit, Dev.To, newsletters like Changelog/SoftwareleadWeekly
- Study the evening before to sleep is very powerful because helps you to remember what you read more easily
- Create a list of stuff to read or study when you have time
- Read stuff from others whom you admire
- Find a way to experiment what you are learning, check the article Don't Learn More, Learn Smarter. A Quick Guide to Agile Learning[113].

I want to quote a phrase from this article Interview candidates with an Open Source background[114] that I liked a lot:

> Professionals with a solid Open Source background do not depend on certifications paid by their companies to learn how to use their tools, or having to change their base tool-set regularly, limited by commercial decisions. In other words, Open Source professionals have made their career supported in specific tooling and associated practices they have chosen and mastered.

[112] https://dev.to/uncagedyou/22-simple-ways-to-learn-faster-3hg8]
[113] https://www.entrepreneur.com/article/341618
[114] https://toscalix.com/2019/12/10/interview-candidates-with-an-open-source-background/

It is easy to spot the real skilled people with knowledge about it, I saw this also on my job. People trust me about specific topics because I know how they work not because I read a tutorial but because I studied the internal stuff and experienced in a lot of different ways. Just creating a repo for hacktoberfest[115] to do 4 pull requests and getting a t-shirt cannot be compared with people who do real effective contribution and improve their skills on high level.
Of course you can create a repo for learning but just doing it in that period of the year doesn't mean that you are a contributor.

Open source skilled people don't have a CV (after all I don't have any) but our career speak for us and is more powerful of a course bought on Udemy. Probably we need to improve our communication skills (one of the reason of this book) but this is another topic.

Some examples of simple tasks that a lot of communities have to help newcomers (also on something that can be difficult, like Git or development with 0 knowledge):

- Contributing to KDE is easier than you think – Phabricator patches using the web interface[116]
- Create a patch for LibreOffice directly in gerrit[117]
- ScummVM is a Magic Box That Runs Classic Adventures[118]
- How I started contributing to the Mozilla Developer Network[119]
- Pull Requests Like a PRO[120]
- So… How do you get started with open source contributions?[121]

[115] https://hacktoberfest.digitalocean.com/
[116] https://rabbitictranslator.com/wordpress/index.php/2020/05/03/contributing-to-kde-phabricator-web-interface/
[117] https://libreoffice-dev.blogspot.com/2020/05/create-patch-for-libreoffice-directly.html
[118] https://levvvel.com/scummvm/
[119] https://eduardoboucas.com/blog/2016/08/17/mdn.html
[120] https://navendu.me/posts/pull-requests-like-a-pro/
[121] https://snehit.dev/posts/foss/starting-with-open-source/

Conclusion

The Contributor Mode for me is the best way to recap everything, see that definition as a motivational process for every task or activities that you will do when you are working on everything you want.
Keeping in mind **why we are doing things is more important than doing the things** themselves often.

One of the matter of the last 5 years in OSS is communication, with the explosion of social networks and new generations of new contributors this is more important than usual. Often in OSS there is the feeling that for some tools you are using like a fax machine instead of website and this is part of the organizational debt of the project that includes the fact that is difficult to do this kind of changes. So, the first step is to not get demotivated because you don't understand how to talk with the community or with someone from the project because at the end you will find a way, it is more important to see if this communication method is still used and not abandoned, like happens often with mailing lists those days.
Like in the Wine examples with GitLab[122] there are a lot of migration to better tools, but still today there are projects that are reachable only with old solutions.

Still push, and you will find a way, it is internet after all, everything is there you just need to find it.

[122] https://www.phoronix.com/scan.php?page=news_item&px=Wine-GitLab-Main-Workflow

Gotta catch ‘em all

In short, how to select **which project contribute to**.
Common question, that for me has an easy answer, but rarely does for newcomers. They are thrilled and excited, but as nerds in a comic books store they have no idea where to start.

Another issue newcomers face, is an Open Source project using proprietary resources to perform processes or activities. This is a common misconception that may hinder the integrity of the project in newcomers’ eyes.
An Open Source project releases Open Source softwares and solution, this doesn’t imply contributors have to use Open Source solutions to reach the goal (an example of this is the use of GitHub or GitLab).

Basically, an Open Source community may have interests/behaviors different from other Open Source communities, that aren’t however conflicting with our expectations, otherwise we won’t be involved or engaged at all. It’s better to engage, and slowly introduce suggestions/proposals, keeping in mind that this shouldn’t happen on our “first day”. Who is going to listen to such a newcomer proposing to amend an established workflow or use different tools while the ecosystem is indeed tried and true and everybody knows how things are to be managed, after all?

When you choose a community you are not choosing its users or contributors, you can only lead the project and the community itself in accordance with the ideals informing them from the start and follow its code of conduct.

Just ask yourself few questions

Are you willing to take part in a project?
Are you feeling motivated and ready for everything?
Are you willing to challenge yourself?
So you are what we are looking for!

You just need to ask yourself that last question: why?
Contributing to a project without having a long term or personal goal is going to stop you from discovering the project itself or from feeling a part of it. Without a goal it will feel as doing homework in a hurry.
The personal motivation is important because it allows you to tackle issue, it’s like a marathon after

all, you may think you can do it but realizing it takes running for 60 km it's another pair of shoes…
I see many people contributing to enrich their resumés, as legitimate and necessary as it is, again, this isn't going to help you feel a part of the project.

You weren't fond of doing your homework at school, were you?
This way you can give your best contribution and mention it in your resumé, but you are losing the real experience and the learning part that is really going to help you, and also you'll lose the experience of working on a big project with other peers.

Often we don't realize how much our contribution is worth – in the Open Source environment and in other niches – nor we appreciate that what we learn by contributing (sometimes also having fun!) may be used elsewhere.
One of the trending topic in self-learning is the transfer learning. Famous people like Elon Musk use this technique, so why don't we too?

Transfer learning: the power to reuse the knowledge learned in other areas or from other experiences/people (even if completely different) in other areas or situations.

Joining a project helps us to improve our knowledge and skills in a different context from the one we think it's natural for us, it might be learning a new language or understanding the logic behind a decision.
In short it helps in strengthening our knowledge by living real experiences instead of "gathering" facts about them.

The experience makes the difference, if it is also followed by meritocracy and satisfaction (normal things in open source) this may help us reach a new point of view.
In this chapter we will discuss the questions we need to ask to understand in what area or project people may be interested, these may be useful also for you when looking for something.
Remember that you need to know how to sell, and that it is useless to propose stuff to an audience that is not interested, it is better to get their interest and engage them.

Which questions you should ask yourself?

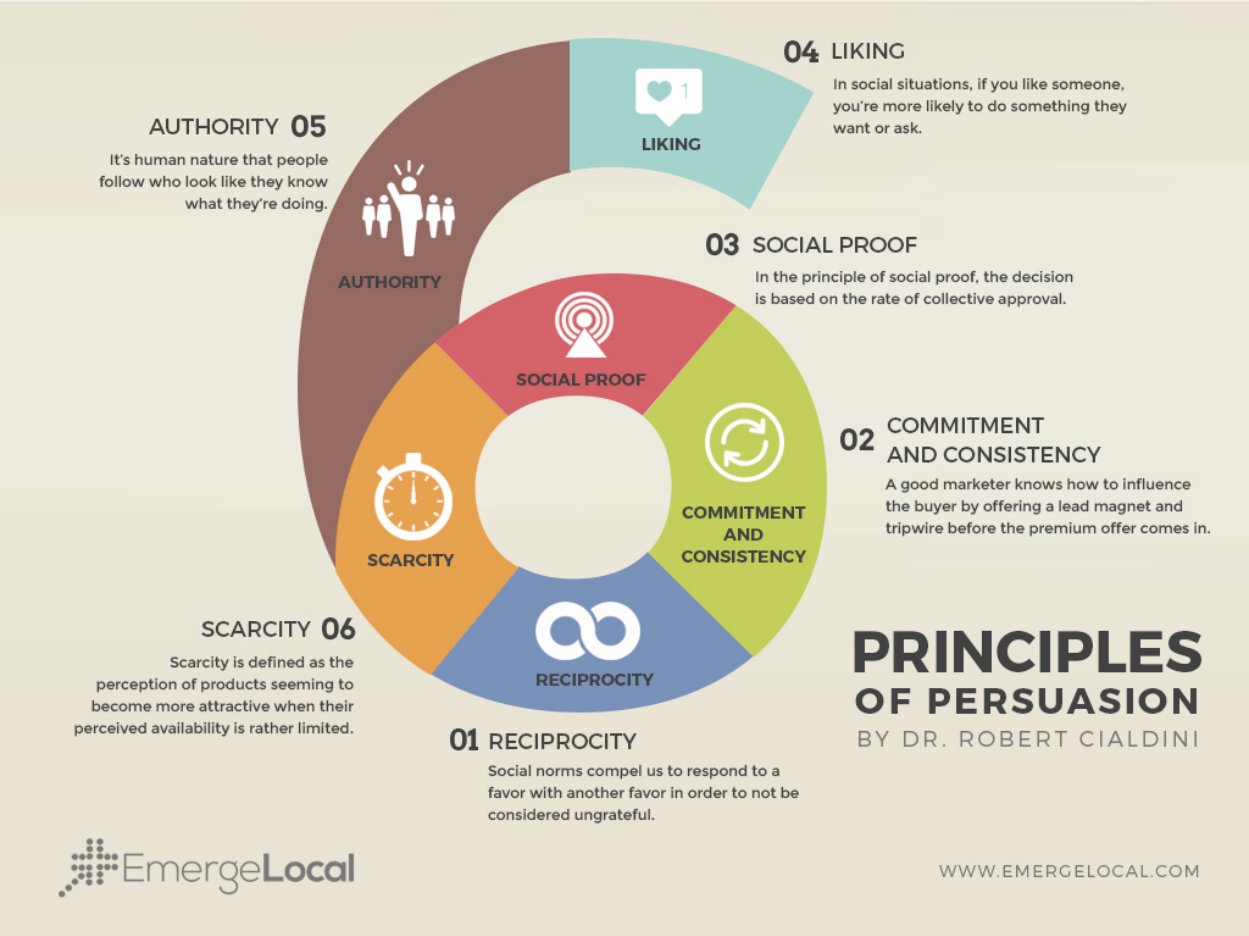
With recruiting we are talking about onboarding new people in your community.
There are different types of recruiting and this manual will show the 1:1 approach but the rest of the suggestions of this document works for every type of recruiting or community life issues.

What is a person's goal when volunteering?

- Opportunity to learn
- Grow in responsibilities
- Contribute with others
- Recognition for his achievements

- Allocating a time slot in their free time

Before moving on, a very good book about persuasion "Pre-suasion" by Robert Cialdini that I suggest you to read (it is localized in other languages too) and uses this recap:



Pre-Suasion infographic by Cialdini

I bought that book while at the airport waiting for the last flight to come back in Italy after a Mozilla's All Hands in US (I don't recall which one specifically) and it has been was very important in shifting my approach to persuading people.

Anyway I wrote the first version of the doc "How to recruit and motivate volunteers" in 11/07/2016 before reading it and it was surprising for me to discover how much I was skilled in marketing and how much I improved in this field. This explains why I was motivated to run as a candidate for the Reps Council too.
Back on track, the next pages are based on this document that is still available on internet (and mentioned in Mozilla Reps resources too).

Think that these points are important to understand how much a project may be easy for newcomers:

- What are the tools to get familiar with the community and for what kind of requests (forum, mailing list, telegram, etc.)? Example: IRC is not easy for newcomers from the UX to access, evaluate other solution like Telegram or Matrix
- What are the tools that the project uses to document itself (manuals, wiki)?
- The most common 5 issues that newcomers have in the project
- 3 tips to contribute to the project
- Why the reader should join the project

Moving to the marketing side:

- Are there big companies contributing or using your project? If yes why?
- How many volunteers do you have?
- How many events do you organize?
- How is the project maintained (private, companies, foundation, etc)
- Interesting numbers or facts that show how your project is changing people's lives

Rules for the interviewer on recruiting

As talkative Italian I defined these rules for an informal chat (1:1 or 1:many-people) or a way to easier/better identify a (future) contributor. Basically you need to find what people are interested in, the area where they live or the language they speak, the background and so on. Like a job interview but with different meanings.

- Ask friendly, you are talking with a probable volunteer so you can be informal, but don't forget that you are looking for a volunteer and not for a professional
- Get informed about the interests of the local people in the area of volunteering you are looking for
- Share a gadget with the interviewer like a sticker to get a friendly approach
- Say clearly that this chat has the purpose of:
  - trying to understand what are the interesting projects for him
  - trying to define what are the local interests
  - collecting information on how to improve the communication
- Speak about your experience as a volunteer: people love to listen to successful stories and yours probably is
- You are "selling" a community with activities and projects so look for the most intriguing and easy ones
- Tell how your community is organized, don't get afraid to say that something is not working because honesty is very valuable, especially if you are working to improve it
- Tell what are the plans for their involvement
- Let the interviewer talk about him/her(self), everyone loves to talk about him/herself so you can find important information for your purpose in a friendly way: a chat (in Italy with gestures of course)

Round of questions for the interviewed

Ask the real questions - https://www.reddit.com/r/linuxmemes/comments/akft99/why_i_use_linux_instead_of_windows/

There are many factors that may shape these questions, like the skills of the interviewer, their professional background and hobbies (student, working, hacker) and why is he/she attending the event, so think carefully what to ask him/her.

The first and second rounds of the conversation can be done together in person or online.
This discussion can be followed with mentoring of the potential contributor after they join the community. This round of questions/chat can be done together so feel free to reassemble it as you want, this is my way of doing that part!

The order of questions is important to understand the needs of the volunteer and what is most interesting to him/her.

First round of chat

- Do you know what the project does for you? Even if you are not using the project itself?
  - Ex: What does Mozilla do for you, even if you are using Google Chrome? It is a simple way to start a conversation about your “product” and engage him/her into a chat about

his/her thoughts. It is a simple way to showcase that he/she doesn't know a lot about the product and get a volunteer interested.

An example is The Document Foundation from LibreOffice that fights for open standard in documents that enable the owner of the documents to be the real owner, since he/she will know how the file is created. This comes in comparison with Facebook where you can publish a photo but you are not owner anymore after publishing.

- What do you like about this project?
  - Ex: Why do you suggest to your friend to use an android device?
- Why are you using that project for the first time?
  - Discover the reason of the choice and find others that can have similar interests
- What other volunteer projects are you involved it?
  - Helpful to find projects that can be interesting to the interviewee. This will also help you to understand better his/her background

Second round of chat

Now the interviewer can talk about the projects that can be of interest for the interviewee.

All people:

- Explain why it's cool to join a community like yours
  - They can improve their skills
  - Meet new people
  - Learn how to work in a group
  - Improve their communication skills
  - Become a project manager, recruiter, translator etc
  - Feel part of a big group
  - Their thoughts are important for the project
- Promotion
  - Share news
  - Do promotion in local events with gadgets
  - Need to know the communities
- Advocacy
  - Like the promotion section but if you would like to advocate for a project make sure you have a good knowledge about the project or a specific part of it
  - Advocating about it in their own websites in their own native language or English
  - Initially a mentor is recommended to review the content

Add also reasons based on your personal experience about issues in your country that can help you to mobilize, like educational topics.

For developers/tech people:

- Promote new technologies and add a few examples about what are you doing as a community about it
  - Online/live courses
  - Local group (national or urban)
  - Hackathons
- Talk about new features of a project that can interest for him/her

This people, like me, are interested in discovering new technologies that can improve their performance on the job or just on a volunteer basis.
At the same time they can be very interested in new technologies, how they work, what they can do and why they are better compared to the other ones.

For sysadmin/high technical people:

Usually a system administrator is not a classic tech audience as we usually think, they are not developers, but they are up-to-date with new technologies especially since that is part of their daily tasks.
They have no time to engage on developers' specific topics, but they love to talk about "changelogs" or news about technology.
Often they are also open source philosophy lovers, or they advocate a lot about security so this kind of topics are precious to them.
Sometimes they don't have social profiles or smartphones, so they are not interested in social media because, simply, they don't use them.

- Talk about the last feature of a technology:
  - Ex: Do you know that Firefox Nightly is a lot faster and support etc.
- Talk about internet freedom issues:
  - Ex: Do you know about the copyright issue on doing selfies etc.
- Talk about a new software that can change their daily workflow
- Talk about how much fun and interesting it is growing skills

For students:

- Talk about the skills that they may learn
- Promote the recognition they may get that can be useful for their studies
- The students coming from the same city can join the forces in their school or universities and work together
- They can meet other students across the country with the same interests

They love to work together, learn and have fun. So you have to be prepared with easy to find and fun activities for them.
This way they can see that is cool working together across the country or globally.
High school students are a complete different case, they have a lot of free time to dedicate but very poor skills/English knowledge.
They are the most difficult to recruit because you need to motivate them with something playful and spend time on mentoring them.

For non-English speaker:

- Which are your preferred activities on the internet?
- What is your job?
  - You need to find an interesting area for him/her, if they are promoters for example, they probably may be interested in organizing an event
- What are the problems you are coming across in the projects?
  - Ask their suggestions, get in touch with them now and not afterwards with an email
- You can improve your English skills contributing

People can be less interested joining because of the language barrier so you have to find quickly something interesting for them and get in touch to convey the feeling that their thoughts are important even though they don't speak English.
DON'T FORGET to leave a reference to the interviewee to contact the interviewer!
Also, they are volunteers, not coworkers, so while everyone can be fit for these roles, you need to be sure there are other opportunities for the volunteers, that you simplified during the interview.
It's true that not everyone can be a volunteer since not all the skills can be used on the current projects the community is involved in, but they can be always useful for promotion or advocacy.

Conclusion

There are a lot of ways to interact with newcomers or noobs (to use a synonym from the videogame world) but often a map is one of the best way to show the community dimension and also where they can find others.
For some projects in Mozilla I developed a map that you can find as boilerplate license in GPL[123] to use as you prefer or get inspiration from.

[123] https://github.com/Mte90/List-People-on-the-Map-Boilerplate

What are the most common areas of activity?

Prerequisite of the activities to propose is the mentioned chat. That way the volunteer is ready to do something, not alone of course, but with your help.
Remember Open Source is not just project promotion but everything that keeps the project alive, contribute in other areas can help your overall view and open new ideas to move further what you are doing and the project itself. They get confidence with your help in the project without experiencing the huge barrier of a link with a list of things to do.
It's not always possible to have that chat but in my experience without that the volunteer will always have problems to understand something of the project – why is this so or how that works for example.

So if you get familiar with someone that already started to contribute don't forget have a chat like this because it may help the volunteer and your approach to the communications in the project itself.
Also,don't be scared by the amount of things that you can find, think small and step by step. Like when learning to swim or ride a bike.
You won't do everything from the first day, but slowly and based on your needs and skills, so let's do the same also for these activities.
(Perhaps) one of the reasons Unix and Linux became so popular is the KISS philosophy.

KISS in short or Keep It Simple Stupid states that most systems work best if they are kept simple rather than made complicated

If this works with systems why can't it work with your tasks?

I will mark with *D* the activities that can have a direct effect on the project and *I* he ones whose impact is difficult to measure but are part of the things people can do even if they aren't active members of the project.
Also, I divided the various areas by level, the higher the level the more experience is required in the project or for the type of activity (basically some activities require skills that you might not have).

Just before reading the next chapter, ask yourself this questions "To live inside this project, …":

- What do you like to do?

- What are your skills?
- What do you like to learn?

The answers to these questions will give you a personal overview of the area or topic that you want to get involved in within all the open source projects around you.

PS: The joy of contributing happens when you see your change/improvement/localization/etc being used by everyone, and they don't know you are the author.
The biggest gratification of a volunteer is the activity he did is used by others!

Imagine your contribution to the project and the effect of other people on what you are doing to get a boost of motivation!
This is also part of understanding what the others are doing inside the project and how it is moving on, where you can help and maybe new blockers.

Just a little note, there aren't only common areas but also tasks that are often repeated or that happens periodically, those are areas where a new volunteer or a contributor can act to improve the amount of time used to dedicated on something more important.

First level activities

Reviewing *D*

This is the simplest activity for a new volunteer that can help you so much (as already said)!
Think about it, you created a documentation, a paper, a design etc and you need feedback before going live and why not ask the new volunteer?
In this way they can make it an important contribution (a fresh eye on docs and the like) for that project and be involved in something very easy. At the same time they learn something about your activity.
Moreover, this task can be performed without a group, enabling to you to prepare something a little more advanced and understand how that volunteer works.

Localization *D*

Start from something that hasn't specific rules to see what are the skills and English knowledge in a group.
The rules of localization can hinder their interest or they may be something they have to learn, and this may slow down their interest.
A new volunteer wants to do something without having to undertake a big course/onboarding to see if fits him.
Moreover, technical glossary may create a problem because the project can use specific terms and the volunteers don't know or understand them yet.

Every project usually has rules or guides about how to help on localizing and is something that everyone can do.

Until you try localization is something easy to say, you have just text to translate. The real issue is to understand the context, how will a newbie read this? A developer and so on. This open many questions and require skills if you want to lead or help to move on this that is one of the most common activities of an open source community. If you are looking for ideas (about how much this is not so easy) Mozilla published their best practices for project managers[124].

Support *D/I*

These activity requires a knowledge of the project, so I don't suggest that initially, but only for the people that are going like it.
Probably a combo will be to translate something about Support.

Testing *D*

This is very important, as is the reviewer, because they can learn about the project and make it something to improve it. It follows reviewer rules.
The difference is that Testing consists in trying to find problems or bugs while Reviewing on the other hand consists in writing a document including all the problems without going in depth about implementation but only to receive feedback.

Promotion/Evangelism/Design *D/I*

Promotion is very important, you can have the coolest project but if the rest of the world is not interested, it will die.
This is a lesson I learned in the startup world, often the most successful company may not have the best project but is promoted better.
Remember, a community without new people or engagement is dead, because people can lose interest quickly and fairly easily.

A community in this field has 3 macro areas:

- Social networks
- Events (that has another section)
- Assets

Social networks are very important today and often they are not open source but you have to go where people you want to reach mingle, so you need to use it.
It is not the purpose of this book to move people among social networks, This approach IMHO is a

[124] https://mozilla-l10n.github.io/documentation/localization/globalization_best_practices.html

very huge failure. If you are selling carrots, it doesn't matter if the people will use them for a salad or a soup, you have to promote it, not to tell people what is the best way to get updated about carrots, that is a very highly forced behavior on your part.

There are a lot of tools that let you to create a team to schedule posts and reply on social networks, at the same time it's important to define rules on how to use them. This because it's going to be an official communication and you need to preserve the brand and avoid conflicts with it.
So you can involve volunteers on suggesting what to share or to do public chat/AMA about different topics.

Instead, assets is more on the organization side, you need graphic assets to share like posters, flyers, logo and so on. This is very useful to better promote your events, and they can still be used after years from everyone.

Confusion

Maybe this is kind of excessive but it's a viral and funny joke in the open source community.
You can't promote your project without a logo and assets to build the website, also posters or other kind of material that may be used for promotion (also from outsiders).

Advocacy *D/I*

Sometimes this area gets hidden but it's very important if done the right and useful way.
How many of you are bored of hearing about the 4 laws of Open Source? And which are the benefits of OS solutions (free and that you can modify it to your needs/likings)? I think we can move on.
The project needs more buzzword to get promoted and to have a chance to fight with other competitors, so let's see some examples:

- Promote Open Standard for documents because there are proprietary softwares used by doctors that are not compatible with one another, so our health data are closed but can save your life
- Promote Open data because it is possible to track and discover new things around you for they are catalogued. Do you know what is the amount of masterpieces closed in the Italian museums' warehouses because there is no space to expose them? Without exposure, you will have no idea about how much one single information can change a lot of feelings.
- Privacy is important because it's possible to identify you even if you are navigating in private mode on the internet due to the proprietary feature of your browser full of bugs.
- Open source doesn't mean it is free, but that the project is under a license allowing specific things. So it's sometimes possible to get the code by paying it for example, like in WordPress plugin/theme ecosystem.

Those are examples of advocacy 2.0 for open source projects. We take an important fact that involves everyone from the newbie to the non-IT aware person, leaving a fewdoubts behind so people will remember the whole point and giving information about something new that they don't know. Also, giving people a choice without being a dictator but motivating them to discover more and not only by talking about your project.
For me this differs from promotion/evangelism because the purpose here is to motivate people to discover more and engage, not only get a gadget or something new to promote.

Event *I*

Often events are seen like amateur gatherings where friends meet to always discuss the same things. This is true, but it's also a way to network, find new people and approach them to promote live. If you want to talk about open source for students you have to do it in a school and you'll have to consider what they may be interested in.
Organizing events and giving talks or hosting workshops is important. It's a huge way to interact with people, understand their mood and motivate them. Humans are social animals that need interaction, and sometimes they need to meet also offline.
The people you engage in events are more prone to be engaged later as volunteers, because they see other people like them or that is possible to contribute because the others are akin to them.

Second level activities

Documentation *D*

I already talked a bit about that previously, but I want only to repeat that it is very valuable for the project brand on the long run and to improve the quality of the project.

Community management *D*

This is a very important task that requires a lot of patience and good communication skills in addition to management abilities.
The community is build around calls or meetings that need to be arranged periodically and documented (there is a guide in the appendix), understanding the various roles for the volunteers, organize the project/resources and recruit and motivate, write reports and proposals.
Also, this is a very important role because usually it is a bridge between new volunteers and the project itself, because people see them more often.
There will be a guide on how to motivate volunteers in the appendix, because also the most active one can have his/her moments of doubt or other things to do, and we need to re-engage them later, or we can easily lose their help.
A little suggestion, don't forget that the community likes to be involved in decisions, because the project is open, so defining ways to let them join in it's very important.

Development/Infrastructure *D/I*

If the project has a software, well they are important!
Without skilled developers the project will be dead in short time, because it will be abandoned and you don't want that.
People with skills, like manage the codebase or how to do the changes in the right way[125], are required to lead it and plan what to do, but also to mentor new people and simplify the onboarding. The development area usually has a very high learning curve to contribute and it is important to have a balanced way to access it.

Also some projects have a different team called Infra/infrastructure/Meta/SysAdmin, the purpose is to keep moving on the official websites and the various tools used by the project from the ticket system to the git instance. They can be developers that create a new theme, specific integrations or sysadmin that keep the server safe and updated.
To get the best from your users is important to offer the best tools, this means that they don't need to be at 100% FOSS projects. Well it will be better but sometimes you cannot do everything in this way, maybe in the future your needs can be inspiring for something new or create new opportunities.
It is important just to focus on priorities, it is more important the tool or your main project in that moment? I ask this because often focusing on new tools or develop it is thrilling for contributors but create issues on maintain them or keep focusing on the goals of the real project.

[125] https://pncnmnp.github.io/blogs/oss-guide.html

Conclusion

The last mention of this chapter is how this activities fit in the structure of the community accountability. Basically every activity or community has various role levels: Leaders, Maintainers/Committers, Contributors.
Those helps on evolving the community and giving different tasks but at the end is always a volunteer, it is just a volunteer with more skills that got this role maybe for experience or necessity. In Open Source the **meritocracy** is always a rule (always in projects where there is no business), where the experience is knowledge/facts based and not financially. We have to consider also the Peter principle[126] "employees are promoted based on their success in previous jobs until they reach a level at which they are no longer competent, as skills in one job do not necessarily translate to another" that can happen in open source also if there is meritocracy as the Maintainer role can have too many duties.

Open source involves a lot of areas where you can find a home for your interests and people like you. The homework that I can give to you is think what do you like most, what will be the first area and the last one that you want to do in your journey. After this, it is time to discover how it really works.

[126] https://en.wikipedia.org/wiki/Peter_principle

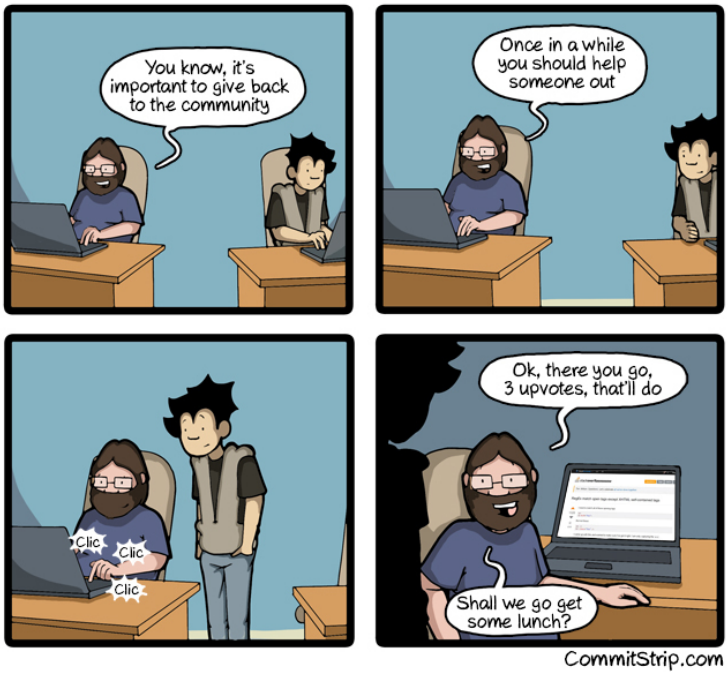
How to find the most important value that you have: Time

This is the biggest question that others ask me when seeing my contributions, or when they are interested in doing something.
I can't talk in your place, but there are different ways to save time and I will show you my way and few tips.
Remember that giving your time is already contributing, next step is to decide in what you are contributing.

Yes, not everyone has time and it may happen that the most qualified people don't have it, so they can't grow in the Open ecosystem. This is something that the project can't change, so they can help on increasing engagement of newcomers in simple tasks and because of that they will find more time for contributing.
If we find time to watch a TV series, a football match, play videogames, eat and so on and so forth, it is because we have an interest in that.

Don't forget to consider your job in that, it isn't something that you can do quickly but you can help on changing the mindset of your workplace to understand the power of it. From specific tools or projects that improve your products to better tools for you. Maybe your company will see it is good that you invest your time on contributing because the company gets what they need and maybe also an improvement of the brand itself (other than more skills for their employees).

So without your first steps you can't think to do something, time doesn't grow on trees. It depends on motivation but also on a good workflow that improves your daily life.



Give back is not a "like" on social networks

My workflow is based on:

- To-do list
- Optimization of tasks
- Use your boredom/break times

To-do list

I contribute in a lot of projects and usually it's because I need something for my job, for my needs or because I want to help.
I ordered them according to their priority and again that shows also why contributing is important because it may improve the quality of what you can do in your job.
With work, I mean a patch to something that I need to use, open a ticket for an issue, translate something or also merely study something and do a new project based on others.
My needs are specially to enhance the next part of my workflow but sometimes also for daily usage of my workstation, like a browser extension or a script to automate stuff.
I want to help: this instead is based on my free time, but how I can merge all of them in my daily life?

I have a wall of post-it, browser's bookmarks of tickets, notes about things to do and so on. I prefer to write any idea to free my mind about those thoughts and focus on something else, maybe I can re-catch them after days or weeks.

I order them based on the priorities and based on performance time and the third part of the workflow. Another point is also part of the project management, check who can do that task and propose it to him/her. In this way you are motivating a new volunteer, becoming his/her mentor (maybe), using the skills of a new volunteer and saving time to do something else.

Learning to delegate is very important and lets you focus on the things that you can do better, in the meantime onboarding new people that (maybe) can help you on that task later.
Another suggestion in case you have tasks that you don't want to do it, instead of doing them later is better to do them as first. In this way you feel more relaxed and free to do the rest of things in queue.

Optimization of tasks

This is very personal and I suggest you to read a few books (check the appendix); anyway for this I suggest to have a look to your tasks. When you do something always in the same way many times a day/week, maybe is time to find a solution to simplify, the machines are perfect for this kind of tasks.
One solution may be to get a mouse with side button to switch tabs or hotkeys to launch programs (or to learn the hotkeys themselves) and so on.
For this I suggest checking the Kaizen philosophy[127], that is basically a way to do a little change that improves something and continuing to perform a lot of tiny changes that in the big picture create a big puzzle completely different from the previous one.

As example after optimizing the usage of the keyboard I moved to optimizing my daily routine, like checking the news-feed after lunch or saving things to read during "news-feed Moments" in the bookmarks during the day. Also, try to dedicate a specific amount of time to emails every hour or day based on the momentary needs and this is another way to aggregate stuff that requires your attention.
Anyway, this can be stupid, any second you save is a way to do something else and make a better use of your time; with tiny steps you can see more tasks that you can optimize like improving the spam filter settings of your inbox or a new feature of your computer. The optimization doesn't need to be intended to save time for open source but time for you to use for what you like but also to be faster.

Use your boredom/break times

In my workday in front of a computer I do a lot of things because between tasks I take short breaks, during them I check my todo list and the one that I want to do (based also of the time that I want to invest).
As example, I do an important task, I deserve a long break maybe 30 minutes or more, so I can perform a task about something I like. This kind of trophy seems like a dog's treat *that was a good*

[127] https://en.wikipedia.org/wiki/Kaizen

boy, a lot of coaching books suggest this technique because it improves your attitude on the job but also your mood.
Getting a break is helpful because it resets your brain so you can start fresh (even for 30 minutes) and work again on the same task. That's also true if you hate it or didn't find a solution before.

They suggest a piece of chocolate, I do something else. Also, doing such breaks during the work is a way to not get distracted or tired working at the computer, we easily switch to different things keeping the attention span working. Of course, I sometimes dedicate the afternoon to something specific,it may be work or an open source task but it's on me to decide based on the priorities what to do.
This should happen to you also, in this way you can better value your time, your work and get more satisfaction from it.

Another way can be defined as a hobby that you want to invest your time, as example for myself for an year I worked to improve my knowledge in other spoken languages. In this way you can appreciate more how you are using your time and define what kind of break you need based on your time availability.

Leadership

It is a fact that veteran volunteers become the reference and usually are getting a lot of requests for help or for their feedback on something.
Those are difficult to plan but it's possible if you leave a spot in your day to check if you have this kind of requests.
Leadership without involvement is impossible, for this reason usually when there is someone that wants to lead and no experience, there are no interests on what he is proposing. This is kind of normal, you need to get the respect from the community in this role, taking care of it by replying to the requests is important.

Delegating

Delegating is not an easy task, especially for founders or people that are involved a lot in any situation. Anyway this is the first skill that any leader need to know to achieve any goals of the project.
To delegate doesn't mean loosing control but instead a way to empower others (and grow the project itself) as we say above i this chapter. As contributor or leader it is important to empower others to create better relationships but most important to do a better team job and create a better team.

The benefits of the delegating can be very surprising because with empowering others there will be probably new opportunities to grow or change something that you never think of.
The real benefits to you are first of all for your health (avoid burnout), understand what are the things that only you can do and what are the area where you are acting as first person.

Understanding this role, tasks and area is important to have some numbers, see bottleneck, open new opportunity to grow and so on. An example is: you have the password for the various social accounts but they are dead because you are busy on fixing bugs or localizing, instead you can create a team or delegate to someone that you trust to follow it for you.
Another one: your project doesn't have any documentation because you want that everything is perfect and the project is not yet ready for your standards, but someone that has better writing skills can do a documentation also for the actual status of your project.

Another important fact about delegating is to avoid gatekeeping. Gatekeeping is one of the bigger issues (for me) in the OS world.
Basically block the access to important roles or onboarding of new volunteers in the project or in specific areas. This create a bad mood inside the community, bottleneck, feelings for a fork, bad reputation of the project, revolution or refactoring and many others. Basically block innovation inside the community and your work because there is no new people that can replace the other ones that maybe are busy or left the project.
In other words a community with gatekeeping is a community/project that is going to an awful ending.

Burnout

About this topics a lot of people with better knowledge than me wrote a lot of books or websites[128]. My experience is basically to understand what are the priorities and define time slot, where after that you don't care except very important things (it is a skill to understand what are those).

Usually burnout has different symptoms: isolation, feeling undervalued, recognition, values conflict, loss motivation and lack of autonomy.
There are some things that you can do quick easy: automatize tasks (I use the rule of 3), understand what others are doing and how, communicate better.

Rules of 3 I don't remember where i found it but I have my own version: if it is a task that I do with the computer like 3 times a day or for week always in the same way you need to automate it. Especially if this task is based on a lot of steps that can creates confusions. In case it is something where you cannot use a computer there are other ways to automatize things in this days with Zapier or IFTTT or if you are a developer you can find some ways.

[128] https://selfcare.tech/

Conclusion

I can't show you how to save your time. It is on you to analyze your tasks and your time and understand how you can optimize it to do whatever you want.
The first step is to learn from others, for example I like a lot to read about how big companies or projects changed (or explain) their workflow like in those examples:

- How WhatsApp scaled to 1 billion users with only 50 engineers[129]
- Shipping a security update of Firefox in less than a day[130]
- The 2022 r/place story from the Italian view (and from bots)[131] - This is mine
- Lessons learned from my 10 year open source project[132]

Or reading post mortem/outage after big bugs or hardware failures:

- Mozilla Add-Ons Outage Post-Mortem Result[133]
- GitLab.com database incident[134]
- Facebook: More details about the October 4 outage[135]
- Retrospective and Technical Details on the recent Firefox Outage[136]

Why? Because in these stories you learn a lot about work organization and how to organize yourself and others with a real story, like a novel.
Also in those cases it is important to save your time and do what you have to do very quickly. Don't forget that Open Source is basically learning from others and using this rule also in other context is useful after all.

Just think about it because in this fast world where everything changes so easily is important to understand time's value.

[129] https://blog.quastor.org/p/whatsapp-scaled-1-billion-users-50-engineers
[130] https://hacks.mozilla.org/2018/03/shipping-a-security-update-of-firefox-in-less-than-a-day/
[131] https://daniele.tech/2022/04/the-r-place-story-from-the-italian-view-and-from-bots/
[132] https://medium.com/@micallst/lessons-learned-from-my-10-year-open-source-project-4a4c8c2b4f64
[133] https://hacks.mozilla.org/2019/07/add-ons-outage-post-mortem-result/
[134] https://about.gitlab.com/blog/2017/02/01/gitlab-dot-com-database-incident/
[135] https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/
[136] https://hacks.mozilla.org/2022/02/retrospective-and-technical-details-on-the-recent-firefox-outage/

Documentation! Documentation! Documentation!

Maybe it's not clear, documenting is very important[137] even if it is boring.
It is so important that after Google Summer of code there is also Season of Docs[138] to support project to help to build… their documentation!
You have no idea how much it may be powerful to document and allow for these resources to be used from everyone.
You need to leave the environment better compared to before, you are the veteran, the one survived to the war about the new UI or the logo changes (I am talking to you Mozilla) and because of your story you need to be the Master.
You have to stop making the things worse, before the new logo change or the new motto to be ready, moving on your hobbits to the next fatigue and making everything better.

Probably too much cinematic but I think you got the point, before performing better than what we have now you have to stop things from getting worse.

This book is based on documents that I have written during the years and also this chapter is filled with things gathered in various resources that I have written for my communities and maybe it can be inspirational for you!

Just as example how FAQ, guide, tutorial, wiki and so on can grow during the time just with a first kick to the ball (if those are open to everyone to contribute).

How to analyze an activity

This is part of a template I prepared to help in defining activities with the feedback from other volunteers, in an objective way and to be later voted from everyone. It is based on the coaching method as workflow but is quite simple:

[137] https://www.mcls.io/blog/encouraging-a-culture-of-written-communication
[138] https://developers.google.com/season-of-docs

- Consider an activity such as organizing an event about privacy and social networks
- What kind of resources do you need? Location, projector, internet, graphic promotional assets
- Did we do something similar in the past? About privacy on traveling. What have we learned from that and what can we replicate?
- Advantage of this activity: doesn't require a lot of technical knowledge
- What is missing in the proposal? An update about the privacy during this year
- What is missing for your audience? To gather a national audience we need local user cases
- Kind of public we can involve? All the ages owning at least a smartphone

This analysis is very important to define the priorities and how to promote the activity itself based on the metrics of the event itself.

How to organize a social campaign

Resources

Every social campaign requires different digital assets to the organizers, here you can find the localization of a guide I wrote few years ago for the PrivacyMonth campaign for Mozilla Italia.

- List of messages to publish on social networks available before starting (already reviewed)
- If they need to be localized, do it 3 weeks prior to the campaign to allow for reviewing time
- Think about some resources to give out in the language you need
  - Avoid paper publications and look for impartial providers like the police
- Usually it's better to share something published from someone else instead of writing a new one
- Resources in English are allowed if:
  - They are technical stuff
  - There aren't resources in your language
  - There is an explanation in your language attached
- Avoid English terms as much you can
- Reference things used in your country to provide a better example
- Remember netiquette
- On Facebook wait 30 minutes for every share
- In the afternoon there is more engagement for what you share online (especially with a photo)
- If you include words like News or Update you can get more engagement
- Graphic assets
  - Provide images only when they are localized
  - Provide posters for offline events
- Talk
  - Provide talks made by the community in Italian in powerpoint (to avoid file formats issues but you can use ODP with LibreOffice of course) to review all together, by using Google Documents (as now Collabora Online is not so performing and publicly available) it's possible to work together more quickly
  - For the event get always a PDF version to avoid issues
  - Evaluate topics that fit better with the local necessities to use along the main topic
  - Create different levels based on the various knowledge for the audience or speakers

Roles

To create a team you need to specify the various roles to work all together and avoid conflicts and issues.

- Talk
  - A person who can do the talk and later plan a meeting to review it
  - Try to not put strange colors and keep the same graphic style
- Graphics
  - A person or more that can prepare the resources in your language
- Social
  - A person or more than schedule the messages
  - Remember the url shortner
  - Keep track of the statistics from the start to the end
- Event
  - A person to reference to for help on preparing the schedule for the various event organizers
  - This team coordinates with Graphic and Talk

Events

A social campaign can also include live events:

- Posters need to be the same with consistent styles only with different information like location and timing
- Talks should be already available from the community/organizers
- Create a schedule based on the audience that you will have, don't get confused by the campaign

Define the issues on onboarding

- Why people contribute to the project now?
- Why they joined the project?
- Define what are the volunteers interests

With those questions we have different Personas to start discussing and see which ones are missing.

- Categorize the volunteer types
- Define their usual workflow

Next steps:

- What you can do to improve this workflow?
- Which new activities can be interested for them?
- Which activities are missing the participation of new volunteers?
- Which are the broken part of the onboarding? (resources on the website, getting accounts to platforms and so on)

Conclusion

Take this chapter as inspirational for you to start writing documentation about community management but also on other things. Write the most stupid things too but write them, the next one that is going to replace you probably won't know everything.
Instead if you are interested on discover how to write documentation from the language or the style there is this guide by Google[139].

[139] https://developers.google.com/style

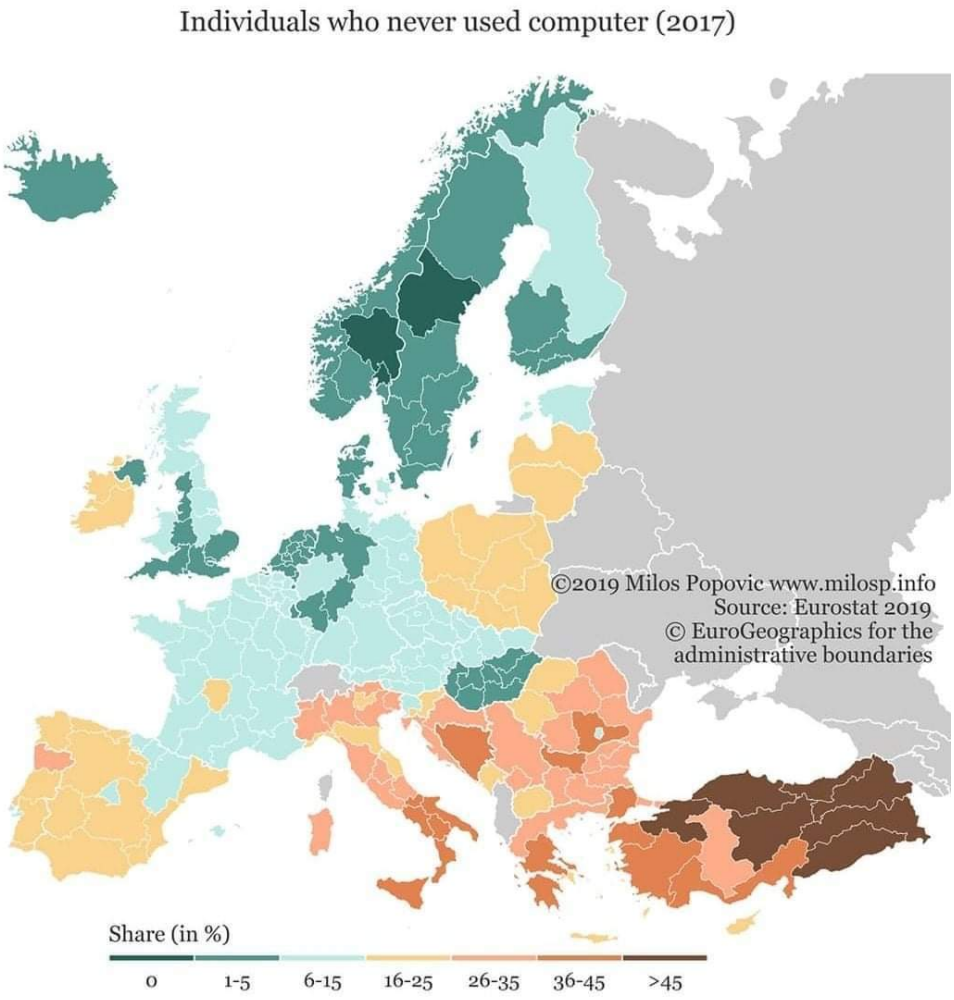
Conclusion

What is the future of the open source world in the next years?

Open source was a scary topic years ago, they used to see you as an amateur. This is fake news today, it is also that but not always. Only because it is free doesn't imply that it's an amateur thing but it can be a good alternative.
Do you know Volvo invented the seat belts but the patent was set up as free to use for all to improve the security for everyone?
I think we need people to discover how the open world changed everything already. For example, we could grant them the opportunities to participate during a time slot in their working hours.

From the routers in every house to connect to the internet, that are cheap because of open source technologies allowing internet to spread everywhere and so on.
We as people need to let the rest of the world be more conscious for what there is around, as we are doing with climate change.
Take as example the power of the **Open Data**, that in some countries are unknown or discouraged from the public governments (I am pointing my finger to you Italy) and how much they can change the approach to the big archives that every country has.

The future is "We need more conscious people in all the topics", with internet we have the knowledge within a tap's reach (not only a click with the mouse) but we are not using it the right way. We need to consider the idea that they don't know something and need to understand what they miss, in this case why their using something and if there is something better for different reasons.



Individuals who never used computer (2017)

Basically we need to move on those points (explained in the next sub chapters):

- From “Let’s migrate to Open Source because is free” to “Let’s migrate to Open Source because it offers more features”
- Evolve the concept of the license
- Be more user-friendly
- Get better promotion
- Open source doesn’t mean release the code

Use Open Source software because is better not because is free

Just this title has some issues because Open Source != free, take as example WordPress ecosystem or other CMSs where additional software is sold but the code is public (and not obfuscated/minified).

A software can be open source but this doesn't mean that it's free, it needs to follow the 4 freedoms that don't mention the money factor.

We as community and users need to move on the open to this new level, it isn't anymore a matter of money as before. Taking as example LibreOffice having to compete with Office 365 that is a cloud service sold to public governments and is cheaper than Office suite to install in the computers.
So IT departments prefer Office 365 because they don't need to take care of supporting it, installing it and so on (welcome to the cloud baby) but this opens the door to other questions. Where are the files hosted? In what country? There are secret/privacy information on them? If I cancel the subscription what it's going to happen?
Open source lets you run software in cloud, but we need to change the approach on promoting it because the old slogans don't work anymore.

This was the way people abandoned Internet Explorer, not because Firefox (the biggest competitor at the time) was free but because it was better, as Media player against VLC, WinZip against WinRAR (everyone used it without buying it), WinRAR against 7zip etc. too.
Now people don't think to these tools anymore if they have better ones (biggest example Firefox Vs Chrome or Chrome Vs Chromium).

This is also part of the fact that big companies and a lot of projects in the world use open source components without giving anything back, no money or other types of contribution. The biggest disadvantage of open source basically, but that it's not the right way to work in the open source world properly. Those days there was a lot of discussions about funding an open source project or the maintainer itself, it's not very easy but is something that the users should ask every time they use something for free.

The Open Source Contributor Index[140] it is a practical way to see the biggest company contributors to the ecosystem and are leaded by Google, Microsoft and Red Hat.

Evolve the concept of the license

It is cool to get the source code but is quite useless if it is not possible to run it. It is like when I got a Blu-ray DVD as birthday gift but I had only a DVD player that wasn't supporting it.
Today with the cloud we need to understand that the code access is enough to be open.
Frank Karlitschek[141] (the creator of Owncloud and Nextcloud) explained very well the issue. The 4 freedoms were created when the computer in the home was the unique way to access a software, now with cloud and mobiles there are different kinds of resources. Access to the code was enough to get Digital freedom, but now it isn't anymore. Our data are not anymore our files but also our interests, our photos and so on.

Also, the Open Source Initiative is evolving[142], and now the membership is not only paid, in this way they tripled their members (500/2000) and improved their workflow on evaluating licenses. To reach more people and improve the awareness it is important to go out on our bubble.

[140] https://opensourceindex.io/
[141] https://karlitschek.de/2019/08/open-source-if-more-than-licenses/
[142] https://opensource.org/sites/default/files/public/2021-OSI-AnnualReport-final.pdf

Open source is for me when:

- The code is open source
- There is a community leading the project (behind a company too, like in Docker, Kubernetes, or Arduino)
- There is no Dictator (I don't mean the Benevolent Dictator for Life)
- All the files to use it are provided (like Machine learning models as Mozilla's DeepSpeech with Common Voice)
- Everything is documented and following a standard
- Release management is not happening at the time of major releases only with a cadence of 3-4 months (or less) (people need to be aware of the changes and to be able to deploy without any hurry)

Benevolent Dictator for Life Or the BDFL is a one person that lead the project future, usually is the founder/creator of the project. Also this role has the last voice on every discussion and in some projects this is important because helps on closing discussions or getting a decisions and avoid too much flame wars. In Linux as example is Linus Torvalds, in Python was Guido Van Rossum, Larry Wall for Perl and so on (they created the project). The point to say Benevolent is because as it is open source it is always possible to do a fork with new leaders if there are big issues, for these reasons is a Good Dictator that act only when needed.

User friendly because not all the people love the command line

A common problem years ago was that open source software was very ugly or difficult to use even if powerful.
Now that has changed but not all the projects are offering user friendlyness, this is a problem that requires more people testing and also more UI designers helping with that.
The Open Source Design[143] community is the first example, also KDE with their Usability & Productivity[144] newsletter and Mozilla with weekly updates from the nightly version[145] or The Document Foundation[146] that often release stats or updates about the projects.
User friendly is not only a better UI or cool logo (that changes every few months, yes Mozilla I am talking to you) but informing the people about the updates during the cycle to involve them more.
Other projects involve them more because they offer more news and people can *gossip* like a TV series, well humans are simple sometimes and is something that the open community can replicate.

Communicating better and being more friendly in everything (the purpose of this book indeed) are the future.

[143] https://opensourcedesign.net/
[144] https://pointieststick.com/category/usability-productivity/
[145] https://blog.nightly.mozilla.org/
[146] https://blog.documentfoundation.org

Communicate better to fight who has the money to sponsor everywhere

The legend that "if it is on all the billboards it means it is good" is not true. The high competition in the digital world changed everything.

We can promote it without money, if we make that together and if the project provides the promotional materials.
In this case not graphics assets or gadgets but data to share. The scandals of the last years with social networks have exposed the fact that data are the new currency that if used the right way may also change an election, so let's figure out how to use it to promote the open philosophy better.
Not only the machine wants data but also people because we are in the world where facts don't matter and only real data, a lot of it, can change this bias.

Open source needs more marketers, a sector that often isn't involved in open source because they don't care (maybe) but it's a fact that we need them.

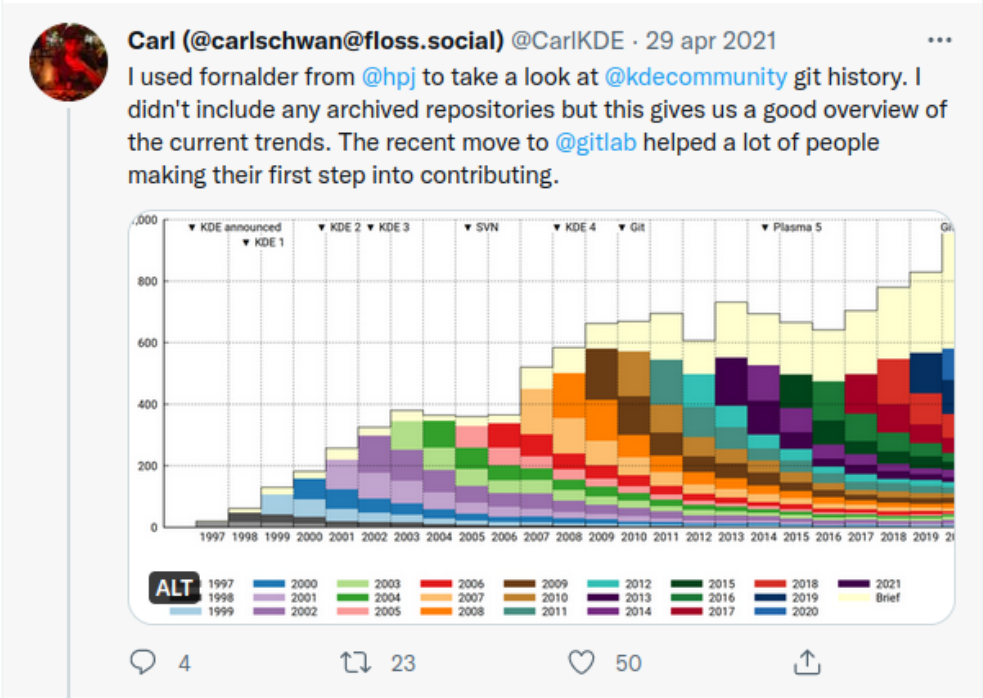
The code is not enough anymore

Today if you ask, to someone that develop or is involved in IT, "What you will do with your project if you have to release it open?" the answer will be "just a zip of the files put online".
As we saw in this book Open Source doesn't mean just release the code but let people to use it and also discover it.

A real example of this is the release of the source code of an indie game, VVVVV that was announced as "VVVVV goes Open Source" but the project didn't had a good license so it was forced to change the announcement with "source code release". There is the original blog post announcement[147] with all the blog post comments but with a research is possible to find also various tweets and discussions about this misunderstanding.
So the code or the core of the project publicly available is not enough because you can only read it, as the legal definition of that is a proprietary project. Without a license that explain how you are allowed to use it is not possible to do anything. Think about of Doom or Duke Nukem open source releases and how they changed the world, also other games that released piece of that letting creating of new games or create new technologies.

The other point is the promotion, with GitHub/GitLab a new issues awaken, someone start a project, release the project with a license and so on. At a certain time the owner is no online anymore or interested on working with that and the project became an abandonware. So a new developer take it with the creation of a fork, the problem is that for the rest of the world maybe this new hope is unknown. All the links and information refer to the original that is abandoned so no one will find the fork. Like a Jedi need to master the force and knows all the aspects of that, also with a release public you need to be ready about what means.

[147] https://distractionware.com/blog/2020/01/vvvvvv-is-now-open-source/



Offering also the best solution to involve and contributing is important to succeed - https://twitter.com/CarlKDE/status/1387754944678215682

Often the best solution is the one that the maintainers or veteran think it is the best, but with new technologies there are new solution that simplify various task that for new people can be very difficult to do.

Personally when I find abandoned project I ask for updates or looking for a new maintainer but often is very difficult to get an answers. Pinging on Twitter or other social media (not everyone publishes their private email address) so the joy of contributing or access to nice things crash against a Death Star, that is the oblivion of the abandon.

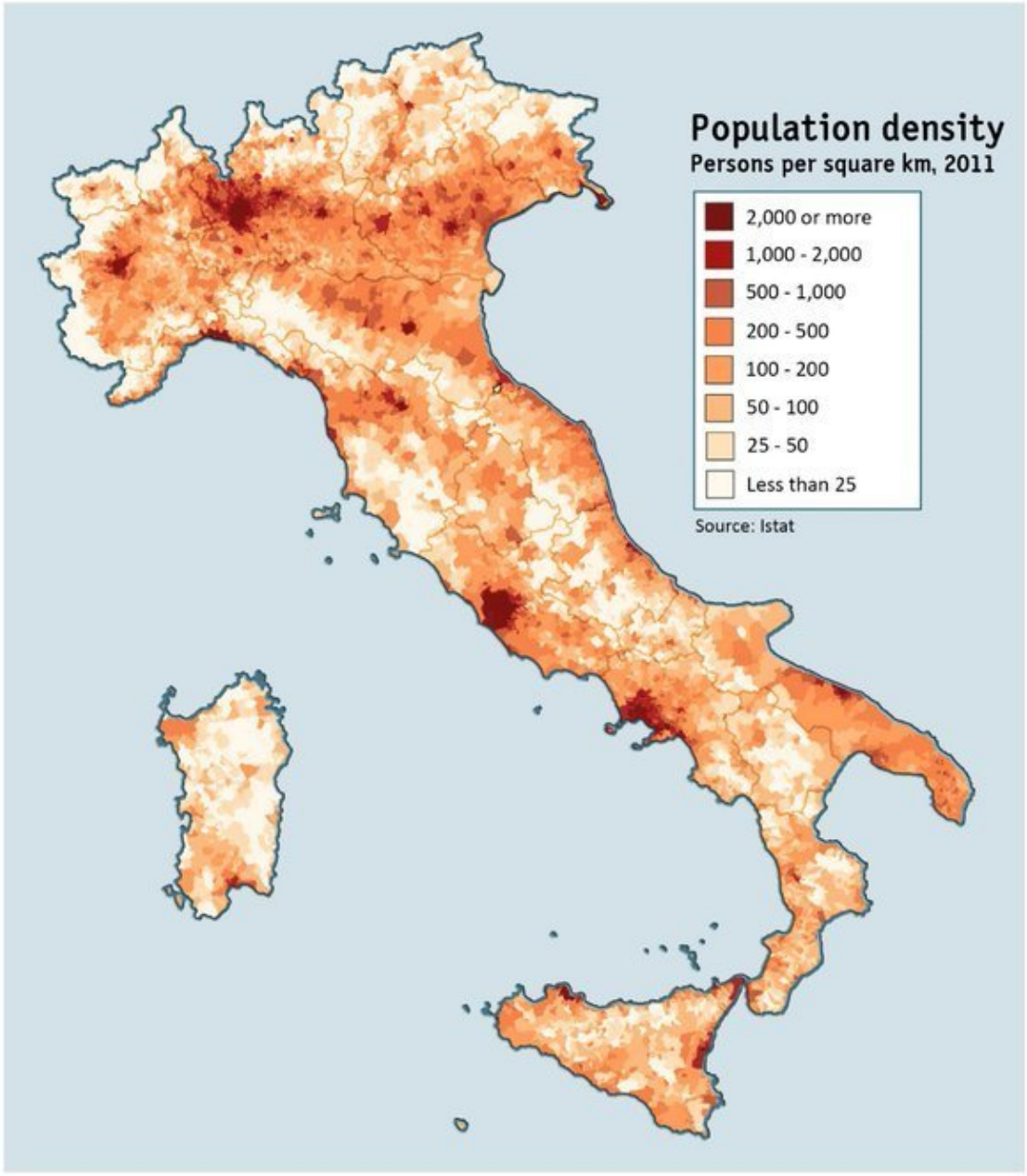
So we need to evangelist better this fact and explain that the code is just a piece of the brand and the meaning of the project.

I don't want to forget also that FOSS is not just promotion. With fediverse and new people discovering this world I see more people that want to do promotion, but it is something that we need after all? In this book I talked a lot about communication and promotion, at the same time there is promotion that change things and promotion that is useless or not working.
The project need it but often is more important instead to be one of the fan in the stand to be the player in the game, maybe not the player but maybe the coach or the doctor.

The Italian case

I am writing this section in English, but is focused on the Italian situation about digitization and awareness about digital matters.

In Italy, the Covid period was one of the biggest reasons for upgrading or understand that digitization is something that improve the life/work quality (like in other part of the world). The real difference (to me) is that involved a lot of the public administration that was always very slow on updating itself but also on the knowledge of the public employees that now were forced to learn a bit about how to use the computer, not just to play Solitaire.



Italy is one of the most decentralized country in the world

As you can see this is the real situation, with people that is not gathered mainly in big cities like it

happens in other countries. This open to various discussions and different situations about how to evangelize or digitalize a fragmented country.

Open data are matters of various association as they are not easily public as digital rights for Italian government propriety like the Colosseum or also The Good Lobby project[148] for sign petitions about various topics from privacy to digital rights. Let's split in two different paragraphs.

Open data in Italy[149] have various issues from the license and the quality of them (when they are available), in fact DatiBeneComune compaign[150] with 58110 signers and 287 organization works on improving this with a monthly newsletter that will keep you updated about what they are doing. The Government now opened the discussion to the Open Data guidelines[151] that is like the first time that something like this is happening (at same time the bigger Italian association for Open Data promotion written a guide about legal aspects on the topic[152]).

The Freedom of panorama in Italy[153] doesn't exist even though we are famous for tourism. Based on a 1941 law we are locked to a definition that says that photographic reproductions of public space are allowed if not create competition against the economic values of the object. This creates issues on Wikipedia side as it is not possible to make certain photos to the Pisa Tower[154] or Castel Del Monte[155] and publish it under a Creative Commons license, if there isn't any act that allow for any of them. The recent European law about copyright didn't help on pushing changes to this law and every year for the Wiki Loves Monuments (global photographic competition) for every local public government they send a request with a list of monuments.

From the software side instead, since the born of the Italian Digital transformation team[156], various things have changed. Not just the Official Forum[157] but also the procedure on what they are doing with GitHub[158] (278 repositories at the time of writing). Including a webpage Come Partecipo[159] to find projects to contribute to, that is based on What I can do for Mozilla[160].
There are public projects like templates for WordPress, Guidelines (with web version on ReadTheDocs), mobile apps, SDK, design kits and others.

Another innovation is the new role on INPS (the national institute that manage the italians money retirements) with CHief for innovation that write a yearly recap of new things[161] on his blog also with a bit of nerdiness. I can remember that they launched their mobile app as OSS[162], a new platform to report issues[163] and many other things.

[148] https://www.thegoodlobby.it/campagne/
[149] https://dati.gov.it/
[150] https://www.databenecomune.it/
[151] https://www.databenecomune.it/2022/07/18/databenecomune-ha-scelto-di-prendere-parte-alla-consultazione-sulle-linee-guida-open-data/
[152] https://ondata.github.io/aspetti-legali-opendata/
[153] https://en.wikipedia.org/wiki/Freedom_of_panorama#Italy
[154] https://en.wikipedia.org/wiki/Pisa_Tower
[155] https://en.wikipedia.org/wiki/Castel_del_Monte,_Apulia
[156] https://teamdigitale.governo.it/en/
[157] https://forum.italia.it/
[158] https://github.com/italia/
[159] https://come-partecipo.italia.it/
[160] https://whatcanidoformozilla.org/
[161] https://vincenzo.me/2022-kamehameha-inps/
[162] https://github.com/INPS-it
[163] https://medium.com/developers-italia

This was something that 5 years ago maybe for a Hard Sci-Fi author was impossible to think of. Like also the fresh news that Inps, Inail and Istat are working together to create in 2023 a unique software house for their need[164] (INPS is the national institute for retirement, Inail the national institute for work injuries and Istat the national institute for statistics).

A tiny things to remember is that with Covid in Italy was defined a new type of work `smart working` with the term in english that as usual for italian language doesn't have any "competitor" in other languages. The difference with remote working for the Italian law is that Smart Working you can go to the office sometimes instead of remote working that is something stable. An example of our technology weirdness…

Two things are very interesting for the purpose of this book about what they are doing.

PublicCode.yml standard

> A metadata description standard for public software that is easy to use both for developers and people with less technical background in order to make the software developed by Public Administrations and Public Organizations easily discoverable.

In few words, a file that you can put in your git repository that is like an identity card that allow to index it with references to the maintainer and dependencies.
The idea is to improve the discoverability and if that project is suitable for a public administration and to use the standard also in other countries.

Catalogo del software di Riuso

That in English "Reuse Software catalog" that Italian public administration can use it. Like every big organization not only departments don't talk to each other, but also cities or region, but they have often the same needs, so why develop the same software multiple times?

This is the idea at the end, a way for public administration where often managers have not so much knowledge on IT to find software already available and open source developed by themselves or by others like LibreOffice for example.

This project uses the standard we saw to fetch the software list, but the website is more friendly as it includes more information, not just a description but people that you can reach to ask questions. It is something that for a bureaucratic organization is important and that in OSS is often ignored. After all, if you are maintaining a project you don't want that people reach you for questions about if the software can do this or that, or if you can implement something just for them. Especially if it is someone that maybe will pay you…

This is an example about how to improve the awareness for organization that is not like the usual software alternative list, because it can give more information that are valuable to the people in charge of decisions.

[164] https://www.wired.it/article/inps-inail-e-istat-stanno-fondando-una-software-house/

In a first world country there are limitations or issues like elsewhere but it is important to see that internet, public awareness and young people are changing the situation. Also, if the Italian case[165] seems that in the public government the people under 30 years old is the 4%. Especially in School institution the age average is the 58% is over 50[166] but in Ocse countries is the 35%.

I know people in Italian universities that still asks questions about Floppy disks to students of psychology for example or that teach using computer with very old operative systems because the professors don't know how to use the modern ones.

The open source awareness let us spread all those ideas and change the situation, takes a lot of time, but gathering various pieces we are getting something. To do this changes is not required evangelism, but also facts like what I am saying. Not just "Ehi that city migrated to Libreoffice" but people that do things, develop software and explain things in a way that the listeners can understand it[167] (Like the Italian guide by the Digital team for managers to understand OSS and how it affects them). Maybe with a catalog with unusual information or showing the issues on various laws.

Thanks

As I already said, I decided to write this book because I wanted to gather all the things I documented over the years, my thoughts and share my experiences.
My experience level couldn't be as it is now without the help of a lot of people:

[165] https://profilo.forumpa.it/doc/?file=2021/2021_FPA_Data_Insight_Lavoro_pubblico_2021.pdf&confirm=yes
[166] https://www.truenumbers.it/eta-degli-insegnanti/
[167] https://docs.italia.it/italia/developers-italia/guida-sviluppo-gestione-software-libero/it/2022-05/index.html

Scasciafratte's last family trip in 2015 in Milan

- My family because they taught me that "A lot of people talk about how to do things and very few on doing them". Without this imprinting, and without my Catholic faith too, is very difficult to imagine my involvement in Open Source. As humans, we have the power to change the world around us, complaining is right only when people eat pineapple on pizza.
- My collegue and associate, Eugenio Petullà, that allowed me to be free during work because what we do for open source is important and brings value to our web agency and our skills. I know that he uses Fedora and Gnome and I use Debian Sid and KDE, but I am trying to ignore this problem.
- Edoardo Viola, leading in the past with me the Mozilla Italia community, who helped me on confronting with others and cooperating in the leadership of a community too. He is a professor now and I hope to get a good rating for this book, but please no oral interview!

Gabriele, Eugenio, me and Edoardo at Fosdem 2017

- Mozilla community for all the things that they taught me, for all the things that I did there, and all the opportunities that I got. Just to mention few people: Michael Kohler, Konstantina Papadea, Ruben Martin, Elio Qoshi, Andre Garzia, Michael Ellis, Anthony Maton, Yofie Setiawan, Irvin Chen, Christos Bacharakis, Tim Maks Van De Broek, Ioana Chiorean, Henrik Mitsch, Ankit Gadgil, Francisco Picolini, Brian King, Havi Hoffman, Jean-Yves Perrier, Catlin Neiman, Emma Irwin, Alex Lakatos.
- Mozilla Italia community members for their involvement in the community itself and for what we achieved in the years, together: Sara, Saverio, Giovanni, Edoardo, Eugenio, Michele, Simone, Damiano, Gabriele, Stefania, Aronne, Niccolò, Francesco.
- WordPress community, even with all the issues I have with the actual project management I am still involved because I trust the people more than the project itself (now).
- To all the other people in other communities that it's always a pleasure to meet and chat with: Italo Vignoli, Enio Gemmo, Giordano Alborghetti, Gabriele Ponzo, Gianfranco Buttu, Roberto

Guido, Marina Latini, Saverio Giulio Malatesta, Emiliano Valvassori, Dario Cavedon, Fabio Pietrosanti, Luca Martinelli.

- All the reviewers for their time and lots of patience on finding English mistakes and suggesting new things that I forgot: Rizki D Kelimutu, Marco Lombardo, Elisabetta Carrara, Konstantina Papadea, Michele Nasti, Takeshi Hamasaki, Stefano Cassone, André Jaenisch (twice), Valerio Bozzolan.

Books that you need to read

- https://github.com/OpenTechStrategies/open-source-archetypes[168] - "A field guide to open source project archetypes" a study paid by Mozilla about the various "business models" that exist in the Open Source ecosystem. An overview about the ecosystem and maybe on something that you don't know.
- Open Source Motivations[169] - A Mozilla study with 30~ volunteers of 2016 (I am one of them) "What motivates contributors to give their time & skills to open source projects? A qualitative study of the motivations and experiences of Mozilla communities members."
- Mozilla Reps Leadership Resources[170] - A collection of resources to read about community management and leadership.
- Open Source Guides For The Enterprise[171] - by Linux Foundation - A lot of information to move your Open Source knowledge to the enterprise level.
- "One Small Step Can Change Your Life: The Kaizen Way" by Robert Maurer - The Kaizen method may help on changing your workflow and being more prone to get better results.
- "The lean startup" by Eric Ries - Following the Kaizen method is very helpful to understand how to structure projects and to get better results in less time.
- "Pre-suasion" by Robert Cialdini - I already talked about it in the book but is important to understand how to talk to people and convince them on what we are interested in.
- "The Tao of Coaching" by Max Landsberg - The first book I read about Coaching after an invite by the Mozilla community managers inside the Reps program.
- "Thanks for the feedback" by Douglas Stone & Sheila Heen - Feedback are important to grow and learn but asking for them in the right way is something that we should know about.
- "How to Win Friends and Influence People" by Dale Carniege and the new edition for the digital era - It is the basic about human relationships, talking and engage better with people.
- "Rebel Code: Linux and the Open Source Revolution" by Glyn Moody - The story about Linux and open source with interviews.
- "Hyperfocus" by Chris Bailey - How to be productive, avoid distraction and use better your time every day.
- "Thank you for arguing" by Jay Heinrichs - A manual about rhetoric and communications tricks to empower your thoughts and proposal.

[168] https://github.com/OpenTechStrategies/open-source-archetypes
[169] http://mzl.la/whyopensource
[170] https://wiki.mozilla.org/Reps/Leadership
[171] https://www.linuxfoundation.org/resources/open-source-guides/

Other resources

- OpenSource.guide, a GitHub official project with a lot of guides about Open source, contribute and so on[172]
- Producing Open Source software, a book by one of the core contributor of SVN[173]
- Mozilla Contributor Research 2017[174]
- Mozilla Open Strategy 2018[175]
- Mozilla Contributor's Communities 2019[176]
- Uncurled - everything I know and learned about running and maintaining Open Source projects for three decades[177]
- Producing Open Source Software: How to Run a Successful Free Software Project (SVN)[178]

[172] https://opensource.guide/
[173] https://producingoss.com/en/
[174] https://mzl.la/whyopensource
[175] https://mzl.la/openstrategy
[176] https://report.mozilla.community/
[177] https://un.curl.dev/
[178] https://producingoss.com/

Appendix 1 for Community Managers

What is a community manager? A Project Maintainer?

There are a lot of resources about this topic but this is my book, so this is my version of that definition. First of all a Community Manager is a leader, and this role require you need to delegate and find new areas of business ehm of activity.
A nice example of community management in a TV series is "Chuck"[179], the BestBuy is a tech supermarket with a lot of different people, background and their various lives. The main character is the de facto leader of the supermarket so others follow him even if he isn't their boss. Listens to the various people, knows them, knows how to moves and be the middle men with the authorities but keep also different lives.
In our case is not a spy but maybe a father, a worker, comics collector or a pirate.
Another TV series is "Scrubs"[180] with different characters, various roles, different lives but common interests or workplace that have to coexist for a purpose. Everyone with their needs and different behaviors but despite that they move on.
I like the pop culture and the reference to cinema/books/comics world because it easy to use the learning transfer (we saw already it in the book) to them and learn how to do not do things[181] or how to manage a company (check Silicon Valley TV series as example).

Another point is a phrase by Italo Vignoli[182], one of the founders of The Document Foundation and one of the OSI council members, that said to me about the role term Community Manager *"A community cannot be managed because everyone do what they want"*. For me this is true, you aren't a community manager but more a community leader that create a path where others converge because they trust you, not because someone placed you for that role.

Basically a community manager is like a boss at you same level, that is more skilled on other areas like communication, relationship and has a more wide knowledge in areas that you don't know.
Often this role is confused with the Project Maintainer or Maintainer, very common in OSS (compared to community manager). A maintainer is a technical role, someone that works to keep

[179] https://en.wikipedia.org/wiki/Chuck_(TV_series)
[180] https://en.wikipedia.org/wiki/Scrubs_(TV_series)
[181] https://blog.tdwright.co.uk/2020/05/07/9-things-westworld-can-teach-us-about-software-engineering/
[182] https://twitter.com/italovignoli

a software to get fixes, new features, code reviews, define roadmap, that says no, but this doesn't involve at all that should care also the community.
About be a project maintainer there are a lot of resources (also in this book) but there are also other guides like [opensource.guide][https://opensource.guide/best-practices/] or from Linux Foundation[183].

A community is a group of people interested in a specific topic that they are pursuing with passion but needs some direction, some management and leading. There are tons of people that are community managers but they don't feel that this role is for them, and probably they are right. Honestly I don't think of myself as a community manager but someone that want the project living and taking this responsibility so others can do what they want (and me too). A good manager is someone that is honest because isn't afraid of discuss about everything with the others, also their ignorance.

Divide your community members to various levels

As we saw in this book a project can have different areas of involvements or activities and this based on the volunteers availability.
It is important to organize at the best what you want to do as community manager divide (just for you) the volunteers in various levels so based on your needs you know what kind of people to reach. We can define as example in this ways:

- Author/Owner - The creator of a project or the is the leader
- Maintainer - A different role then the above that is doing the hard work of keeping everything work and also checking for issues or new opportunities
- Contributors - People that can work based on their time on specific activities
- Community members - the whole group of people that is moving on the project

This groups of people can have as example different communications or different knowledge of what is the project is doing. Keeping an eye on those will help you to find people "to promote" on different tasks that can require more accountability as example.
It is important also to understand the kind of users you have in the community and their interests, in the Conclusion chapter in the Other Resources section there are some interesting reading about the various type of people that can exists in a community. This study can help you on decide what is better to do it and engage more the people that you have and new ones.

To know better about the various jargon or meanings from the various kind of users, I suggest to you to check this talk by Italo at Fosdsem 2020[184].

Another view can be to look at your member like an employee, you want them happy and that they can be productive. At the same time you want to lead them but also give freedom to experiment and express their creativity[185]. Or you can learn from The Office[186] tv series some tips.

[183] https://www.linuxfoundation.org/resources/open-source-guides/winding-down-an-open-source-project/
[184] https://fosdem.org/2020/schedule/event/user_standing/
[185] https://blog.pragmaticengineer.com/what-silicon-valley-gets-right-on-software-engineers/
[186] https://www.quill.com/blog/workplace-culture/company-culture-tips-from-the-office.html

How to do productive discussions

One of the problem in a community that is easy to change the focus or the point of the discussion, in other words to go off-topic. This isn't a specific problem of the meetings but with any discussion from emails to chat.
My experience taught me that in a chat if you sent 2 messages (not 2 questions in the same message) people will reply to just one, maybe the first one or the last one.

There are 2 kind of discussions productive or useless, as maintainer or community manager is important to get the maximum to save your time, because communication is a time-consuming task.
How you can recognize useless discussions? Check this list:

- When are used a lot of hyperbole
- There is a lot of repetition
- There are tons of new ideas without any clear proposal
- Noise detection
  - Comments from inactive people can be very dangerous:
    * They are not aligned with the project status
    * They are just lurking so don't have any experience of the community life of the project
    * Can act as troll because you just changed their workflow[187]
  - Doers are usually silent and is important to reach them

These are basic rules but you need to give a different weight to the your evaluation based also on the previous section. Every volunteer is important but some can have more relevance on the decisions just because they are alive and participating in what you are doing.

How to manage a meeting

Meetings can be boring so it's important to have an agenda with strict times to be sure not to forget something or keep at pace with the plan itself.
During the meeting take few minutes to talk about the purpose of the meeting (also on the invitation email/communication) and the workflow of the meeting.
Preparing the attendees it's important to improve the quality of the meeting itself and to be proud of what was done during this time, so it is not going to be a boring meeting.
The agenda of the meeting is important, and also the titles of the various moments that help to remember the topics (a nice tip can be to add a prefix on your communication like "Website localization - First steps" or "Event organization 2020 - Venue").
The order too, put on top the topics with more priority, avoid jargon during the meeting and don't forget to leave time to reply to questions and also the last words to the attendees.

[187] https://xkcd.com/1172/

Don’t forget that someone needs to get notes of the meetings for the people that missed it and to write a report, also define a specific amount of time to avoid off topic discussions and be on track.
At the same time can be important to have a moment during the meeting for open floor questions but just define in it on your agenda so everyone can plan what to ask in that slot.

When

The new volunteers need to participate to feel a part of a big team so is useful to do one monthly, biweekly or weekly meeting about the status.
When you are working on a specific project you can organize other meetings with their plan and specific people.
It’s important the timing of meeting is chosen gathering team’s availability, and remains almost unchanges during the iterations to avoid issues.
There are 3 type of meeting:

- Live meeting: meeting people in real life it’s important but you always need an agenda with very strict timings
- Online Text meeting: these meetings have a little agenda of 2 or 3 points and someone directs them. They are helpful to vote because they are very quick and can cather to many people and are easy to follow for new people that don’t need to join express their opinions. At the same time they can be made public to allow many people to follow them.
- Online Video meeting: Like live meetings but online. In that meeting the people can be easily distracted or bored so look on the agenda.

These are rules for all the volunteers, but they are important for a new volunteer!
Because they show you are very well organized and motivate him to join in the next; also don’t forget to say hi to the new volunteers and ask them for a little introduction of themselves.

How to approach a problem/improvement or how to organize a new project/team

In every team there are discussions about new ideas or problems/blockers that need to be solved and there are different tricks to speed up and define a roadmap.

- Try explaining only with words what are the goal of that, Ex:
- Get fun
- Accountability
- Involve
- Recruiting

This points will be your compass on the discussion (also will invite others on helping you because the goal is the same), also a reference or the reason of the plan itself!
Gather ideas with these words to speed up the process (is the first step) and you can do it in one of the many ways already explained.
Also, you need a person who will be the lead of every step, not multiple but one or two.

Start also with few people but clear goals and a clear purpose. Create a group with similar skills in that way they can focus better and follow a usual way to do things for their type of skills.

Also, doesn't forget to classify the various tasks for skills, so is simpler for you to assign it (and get a result), but also find the good first task for newcomers. You always need simple things for new people because this is the part that shows how much is easy to join the project or contribute to it (consider busy people, they want to contribute too).

Don't forget to organize periodic meetings!

Analyze a problem or improvement

After the organization of the meeting the workflow after that will be to organize the priorities and:

- You are in a team, even if you are the leader, so ask for feedback or a solution
- Create a user story or a persona
- Select a precise number of people for feedback
- Ask questions also if the solution is already known
- Nothing is foregone
- Leave of the info about this situation
- Pick few goals and make a comparison between today and ahead
- Don't forget that coaching is different to feedback
- Don't forget that the attendees are not you
- Lead by example
- Say thank you at the end of the meetings and on feedback
- 5 why
  - This approach came from the agile method, it consists in asking for 5 times why in a cascade way about the problem to go deeper in a simple way to analyze later and know the environment when the problem showed up

With this approach you will improve the relation with people and also the organization and became more able to analyze the problems. No more boring meetings!

How to communicate

Communication is the core business of a community where the feedback is the fundamental rule (check also the next appendix about this topic).
Also, don't forget to avoid Jargon or explain it when you use it.
In every communication it's better to follow the rule of 7 communications.

- **Completeness**
  - Be complete in everything in your communication to avoid losing time and allowing you to be more productive, as example define goals.
- **Conciseness**
  - Complete of course but also concise to simplify the reading. More text means more mistakes. As for developers, more code means more bugs somewhere. TED talks seems to have the rule to expose the plan in 12 seconds.
- **Consideration**
  - Don't forget that your audience needs to follow you so it's important text is understandable from everyone of your audience.
- **Concreteness**
  - The object of the communication needs to be complete and clear to everyone (maybe evaluate the use of future tense). Dividing the communications into different points is very helpful. Basically don't go off topic.
- **Courtesy**
  - No negativity, not be the victim and be open. It can help with all of your messages and don't forget to send it at the right time during the day.
- **Clearness**
  - The message need to be understood not for the amount of words or the quality of the language but for the context. Putting the proof before the conclusion is the TED way or inductive reasoning.
- **Correctness**
  - Be sure of everything that you are saying to avoid conflicts or misunderstanding.

"Inductive reasoning is distinct from deductive reasoning. While the conclusion of a deductive argument is certain, the truth of the conclusion of an inductive argument may be probable, based upon the evidence given." Wikipedia[188]

When you have no idea where to start with a communication the simple way is writing a list with every point, prioritize and only then writing your message.
Because you are part of an open project, you need to do open questions to the people too. These open questions are not for better clarification but let the others explain what they mean, because a lot of things are not specifically communicated with words for example.

There are different tasks about communication and for a community managers is important to keep a public communication on high quality standards, to save time but also to be more transparent, create an history, create a workflow and improve the reputation of your project. There are 2 ways for that: gather the common questions of newcomers of the community and create some sort of documentation or keep a period report of what is happening/ed in a project. Those are simple ways to improve the quality of communication in your project that on long term can have a lot of benefits.

[188] https://en.wikipedia.org/wiki/Inductive_reasoning

How to talk with people

- Don’t criticize, condemn, nor recriminate
  - In this way your position as leader is friendly and you can create a different atmosphere where people, even when doing wrong things, can continue to work and experiment
- Compliments, even for little things, are important for the people involved because they see that you care about what they are doing. You need be honest of course!
- You are a leader so you need to be an example for them to motivate the others to do the same
- As leader you need to know about “your people” so keep updated about their life
- Smile because no one wants to see sad people
- Let other people talk about themselves and their ideas/suggestions
- Talk about the stuff that interests the audience
  - As we have already seen in this doc it’s very important to get attention from the audience that because that is what you want
- How to persuade others
  - Avoid discussions to be the winner again. You are a person like the others so be human!
  - Respect the others ideas/thoughts
  - If you are wrong admit that as fast as you can before the situation goes worse
    * Start a discussion about a comparison with your experience that was worst and see if the audience is doing better. In that way are you more friendly and human.
    * At same time in a discussion is helpful to start from a point where both the parties agree, so it is possible to prepare a ground to move further the discussion
  - Be always friendly
  - Make questions because only the people humbling in front of others are human and friendly because even if you are the boss they need help from you. No orders to you people!
  - The reputation even if people are wrong is important so let them “save” themselves so they will perceive you as a good leader
  - Motivate of course, because with the wrong mood is difficult to accomplish something
  - The audience need to be happy to do what are you saying
  - Convince people to agree with you from the start of the conversation
    * This way is very powerful to convince people, usually start from a common topic letting you introduce more topics which initially the others are not interested in. Ex: Chrome spies you a lot about what you do while surfing the net but Firefox does not and at the same time it’s very customizable with other things
  - Let others talk
    * Again let the people share their thoughts so you can find more points of discussions
  - Let others have the idea
    * For example you want a new feature, discuss with a developer about the issue, probably he can improve it and fix it for you
  - Think as one of the audience
  - Challenge with them
    * Competition often can create a good mood and spirit inside the community

How to write reports

Every community to be transparent needs to write report about everything from an event, meeting or a roadmap.

All the points about communications can be used for that but there are more things not to be forgotten to help you write a report that is helpful after a year or to people wanting to join you with no background because often reports are public.

- Divide the discussion into different points
- These points are the important stuff that needs to be separated from the rest for better readability
- For every point write a tiny sentence
- When finished review all the points and start improving on the sentences
- Probably you will see that you need to add more points
- Add photo or external resources to improve the quality of the report
- Don't forget to indicate at the beginning the names of the people that joined
  - Also, remember the compliments
- Divide the discussion on a daily basis to show the progress
- Don't forget to do a recap about:
  - What you learn in the event
  - What are the first step to take
  - What are the problems identified
  - What you like of this event and what are the problems and where there is an improvement margin
- Probably it's to soon for a time line but a draft can be very helpful

Don't forget to do yearly reports, campaign reports or other kind of reports. They are useful to show the status and how much community is healthy, they are easy to share online and improve the project brand.

Why you need a script for your next event

This is something that I learned in Mozilla, with new volunteers at their first public event or talking the first time from the booth.
The big issue was everytime they didn't know what to reply to X or Y, so how can you help them to be more comfortable?
Quite easy, do a list of topics with a recap about the status or description to share before the events, in this way the volunteers have a reference to study, they learn something new and you are more relaxed during the event.
About the topic: it is on you, usually they are chosen from: the most famous, the most discussed on internet, the biggest new features, a common issue and also the frequently asked questions.

Communication channels

Every culture has differences but there are few rules that apply for everyone.

- Facebook groups: they can be a mess because there is no moderation of the texts and can become very spammy
- Facebook Messenger: it's a chat and requires you to add every new person on your Facebook, that usually it's private so not everyone wants to add everyone on it
- Telegram: is the best choice because has bots, link previews, moderation tools, links/media recap, client desktop/mobile and is open source (there are open source addicts that don't want to use proprietary software)
- Hangouts/Hangouts Meet/Zoom/Jitsi: it's useful online for video meetings but remember it allows a maximum of 10 people on the free plan
- IRC: it's the classic solution on open source to chat but for the new generation and for the mobile world it's too difficult to configure and understand
- Matrix/Slack-like: it is a IRC 2.0 that may improve the organization in groups of people and usually a bridge between IRC and Telegram can improve the migration. Slack have problems on onboarding new people because the free plan is very limited to 10000 messages (also on private groups and is very easy to reach that). It is not open source and there are many alternatives without that problem. This tools, like Telegram or Matrix have integrations such as bots.
- Mailing list: like IRC they can be annoying and noisy, but some communities still use and it is not possible to avoid them, but you can get answers more easily as a lot of people are following them
- Discourse or forum: with a modern interface they work very well with notification and other stuff
- Collaborative Documents: they are important to work together on a plan. Etherpad in the open source world are the best, but they are very poor in features like: comments or suggest changes, notification and classic features of a text editor. Usually Google Docs are used, covering all those features but the urls are very complicated and some people don't like that because they are not open source.

How to do a roadmap

Briefing

Objectives and Key Results, is a decision framework to improve the group working and define what is the result that you are trying to achieve.

When you have an OKR (like to grow the followers in 3 months or write a new set of documentation pages etc...) the first thing that you have to do is what you want to achieve personally.
Take an example: at the end of the quarter I will reach out to all the team leaders with a survey

to introduce something new that in the meantime I will work on. The first thing is gathering the resources that you need to move to your final goal.
Without you cannot start on prepare the plan, so an overview about documents, contacts, tools etc. is important to see if something is missing.
The next step is to understand if what you have is enough or need to be improved and based on the resource itself give to them different priorities and assign tasks to other people (as example).

Document Structure

When you have an idea of what you already have, you can work on what is missing.
The usual structure of the document to help you to do the various tasks and the final proposal is:

- Recap the actual status in the document as reference for others (this will help you during reviews as example)
- Analysis of what to change with details and explanation (add also user cases)
- Do a final proposal that is a recap in a bullet list (this simplifies for feedback and for the others to remember the various tasks/difference)

Workflow

When you have your proposal but also during the writing you will have tasks to do, like gathering the resources etc.
The best way is to track everything somewhere and in your plan (that is private to the leaders maybe) write all the various tasks.

From a big task, is better to create various little tasks so you can track your progress more easily and see what is happening.

Appendix 2 for Tech Speakers

This appendix is not just for developers or people that speak fluently Jargon. It is open to everybody that need to talk about tech or need to improve communication skills.

Schema to expose at the best your thoughts

- Explain a choice
  - Using the time as your ally to convince more and more people use the rule: Blame is for past, Values for present, Choice for the future. Write your proposal in this way help you to define in the mind what was before and what you are proposing using as base the real situation. Can be something easy to understand but often during discussion we confuse the verb tense, in this way you are processing and writing the information in the most clear way.
  - Write in 3 steps: describe benefits, show that is easy, show how defeat the other option. Before to show the conclusion you can leave also few options or alternatives.
- Understand the target
  - If you want to sell a washing machine you need to know if the customer is someone who clean plates on his own or prefer to trash them. As you are proposing something you need to define the target or the audience, in this way you can change the language, terms but also explain differently.
- Where is the value
  - People will listen to you more clearly if you are talking using your experience or anecdotes. As humans we love a story, also if we can identify ourself, this way to expose things is the way used in TED talks[189].

Ways to get inspiration (especially for funny facts)

The idea is that every talk should contain part of our life as a person to better explain the topic and why we care.
It is also better if we can add funny facts to add more reality but also a different tone to the speech given to the audience.

[189] https://www.ted.com/

- New users with the technology
- Compensation
- Social media gaffe
- Experience-based learning
- Differences with languages
- Proper planning
- Budgeting
- Confidence
- Comfort zone
- statistics
- Habits
- Asking the right questions
- Innovation
- Expectations
- Marketing
- Mistakes
- Embarrassment
- Productivity hacks that didn't worked
- Talking to customers

How to talk to the audience

If you have never given a speech to an audience the first issues are going to be:

- I have no idea where to start
- I am not a good speaker
- I am afraid of the public
- I have nothing interesting to say

How to fight these thoughts?

- Take inspiration from others!
  - Pretty simple look how others talk even of different topics. Consider interviews of famous actors to understand that is not so difficult too. Watch Ted Talks for example or search on YouTube for talks.
- Start with a simple and tiny goal, step by step!
  - For example draft a presentation and try it in your room
- Be positive about your talk! Without this everything that you will do in your life will be a mess.
  - Don't start with apologies because people don't like them. They are here to listen about your topic!
- Practice is the way, also taking the time to recorded

  - On the long run your recordings are the best way to see where you have to improve
- Fear of the public is helpful to do better
  - Fear is a way to empower us to fight it, to do what we want
- Don't memorize but practice; is better for your "show"
  - Memory is for poetry, you need to speak with your feelings on the sleeve
- Take your time before starting to reorder your thoughts and be more fluid
  - Usually prepare a schema of your points for the slide that will be your talk path
- Take confidence with the topic of course! I don't want to listen to someone who talks about something that he doesn't know, because I am losing my time.
  - People come to the event to listen to you, give them something that deserves their time
- The topics are better when coming from your personal background
  - As per the previous point, every topic that is personal is better to give more trustability about what you are saying
- Put examples in your talk because everyone can follow them and are very attractive and easy to explain
  - It is demonstrated that stories, jokes, graphs and images can be helpful examples to let people remember easily your points
- If you have no idea where to start:
  - Use a question or a quote
  - With an image
  - With stats
- Details improve the quality of your talk
  - With the personal experience
  - With strange situations that can be hilarious or easy to explain and get to the point
  - Statistics are objective and very easy to understand
- Why you are here, you are a human and people want to know why are you here
  - Again personal experience does the difference
- First issues about the topic and how to resolve them
  - People go to events to learn how to do better or discover new things
- Be part of the public
  - Say what the people want to listen and understand your audience
- Keep your tone of voice in check, don't speak at a monotone rate or you'll be boring for the audience
- Eye contact is important, there are different ways
  - Look for a far away point in the middle of the audience
  - Look specific people that are interested
- Don't give never your back to the audience
- Your posture needs to be relaxed and confident
- When you want to do a comparison about stereotypes or experiences never generalize to avoid issues with people that fit in that only for a part and avoid conflicts

Don’t forget that every discussion as 4 points:

- Persuade
- Inform
- Convince
- Entertain

If your talk or discussion has all the points it will be probably an amazing experience for the audience!

These points are not mandatory but helpful to get ready and draft your talk.

Your backup sentences in case of issues:

- If your joke falls, say “Pretend it was serious”
- If you lose your train of thought: “Sometimes silence says a lot, but not this time” or ask the audience “What I was saying?” when someone reminds you “Oh cool, someone was listening”
- If the room is too hot say a joke about the thermostats melted and you can’t change it or if it is cold you can say that you saw few penguins around that want to follow the presentation
- A big noise outside the room “That’s what we do to people who try to escape my talk”

How to write a talk

- Don’t forget to talk about yourself, you are the guru for that topic for the attendees
- Usually the first 5 minutes are very important because the audience is going to decide if your talk is interesting or not
  - So an introduction of the topic can be very important
- Show a concrete example/way to fix a statement
- Be specific when you can and generic to be more serious on what you are saying
- Identify the problems of your proposal
- Start every slide with the important part of the content (it will be easier to remember)
- Think about the audience skills
- Identify as a person with that problem, what do you want to listen to?
  - What you are trying to promote
  - The topic points focused simply
- Understand the mindset of the technology you are talking about
  - Flames are not very good during a live talk
- Create in your mind a boilerplate version that follows the slides and an advanced version, to be used for an audience with more skills where you can use the slides as reference to explain more in depth
- You are not a teacher, you are a person with experience in that topic and you want to show your experience

- Simplify, it's important but don't forget that sometimes it's not the best way, but you can divide the topic into different slides
- The people can forget the focus so act like a showman to gather the attention from them
  - Funny images (I am addicted of GIF animated or meme)
  - Quotes are difficult to remember or to understand for an audience that doesn't know that quote
- When you write a new talk try to understand if the slides order is right or you are missing something
- The practice is very important
- The title talk needs to be interesting and curious
  - Use a title generator to gather inspiration
- Empathize few words help to conveying a message
- The slides need to be useful also without attending the talk itself or remembering it
  - Many slides with only images in that case are not very useful
- Last slide for question with your data like email and a link to see the slides again
  - I also have a page with the list of all my talks, in that way they can find also other things

Improve the talk

- A talk without improvements it's not a good talk
- The world evolves with new technologies and things so it's important to update it
- If you get often the same questions in your talk, add that information in the talk itself
- The slides can't change but the way you explain the things can change based on the audience
- Improve the examples, often they need to be updated because for example after a year it's not working anymore

Tips

- The talk need to be done in standing up
- No stop words, like ehm uhm, but in that case stay silent
  - o exercise talk about something for 30 seconds and register it and listen to it. Repeat until you will stop to say them. Do it for different subjects for weeks every day, this can help you speed up on building sentences on your mind.
  - You need to hear on recordings when you are saying them, in that way you can prevent them, is difficult but on the long run you will see the effects
- Try to practice the talk before the talk
- Check all the slides and the content before the talk
- Arrive in the room 30 minutes before your talk to check if everything is fine
- Train yourself on improvisation, try the Battledeck (or Powerpoint Karaoke) with friends. It is a slide deck with random images where the participant have a specific amount of time to talk about the same topic.

How to write a tech article

Every tech speaker sooner or later will write a tech article or manage the personal blog.

Let's see which are the best rules for this task (helpful also on preparing a talk).

- The title need to be intriguing
  - If you have no idea for a title, use a title generator like http://www.title-generator.com/index.php/best-online-title-generator.html[190]
- Start with a question, an image or with stats
- Start the paragraph with the important topic and later on explain it
- Divide the content into paragraph to avoid the "text wall effect"
- Break the line when a specific point was completed
  - It is helpful to avoid repetition
  - On the same block the same concept
- Conclude with a strong image to enforce your point (memes usually are the best option)
- What is the purpose of the article?
- Order the information using this schema:
  - 1st point:
    * tart with the same knowledge ground of the reader, example: your topic is about a new javascript library, the reader probably now has no idea about what to do so it is better to start by explaining what was required to do before the introduction of this library
    * Put at the top the important things to explain later
    * Start with a simple example or point to be comfortable to the audience
  - 2nd point
    * Talk about news, details or with advanced view
  - 5th W order
    * 1st point
      · What, When, Where
    * 2nd point
      · Why, How
- When you want to do a reference about something already written few lines above do a little excerpt of that to avoid to the reader to lose the reading point
- Every item of a list need to start with a different word
- Avoid text block with a lot of text, better to split them
- Metaphors or analogies are more easily remembered by the reader, especially at the top
- When you are using a technical term explain it on the first use
- Be original with your content!
  - An example can be to start from the conclusion and explain with the content. In that way the reader already knows what will be the focus and can be more interested

[190] http://www.title-generator.com/index.php/best-online-title-generator.html

- Every text need to explain at maximum only 3 ideas and develop the surrounding content
  - 3 ideas are enough because the reader will more easily remember them
- Remove the excessive things that are not important for your content
- Put “Sources” or external link improve the quality of the content (and a bit the SEO)
- Don’t forget to ask feedback on the content
- Don’t forget that you are writing to inform the reader, so what is missing to him to understand the content?
- The use of paragraphs let you do a brainstorming based on these blocks, and every block is helpful to reach the conclusion
- Don’t forget that example case are very easy to remember to the reader

Be ready for a CFP

- http://scottberkun.com/2013/how-to-write-a-good-bio/[191]
- http://lucybain.com/blog/2016/conference-proposal-ideas/[192]
- http://scottberkun.com/2011/speakers-checklist/[193]
- http://www.inc.com/sims-wyeth/how-to-capture-and-hold-audience-attention.html[194]

PS: Keep a copy of your bio for the future, it’s boring to write your life every time.
Three version as one sentence, medium version and long version.

Also, refresh them every year to update about your actual interests. Don’t forget to check Developer Advocacy[195]!

[191] http://scottberkun.com/2013/how-to-write-a-good-bio/
[192] http://lucybain.com/blog/2016/conference-proposal-ideas/
[193] http://scottberkun.com/2011/speakers-checklist/
[194] http://www.inc.com/sims-wyeth/how-to-capture-and-hold-audience-attention.html
[195] https://developer-advocacy.com/

Appendix 3 for Mentors

Open source is not for everyone

There is this huge misconception that because it's free it's for everyone. Excluding the fact that open source doesn't mean that it's free this is basically wrong.
People are interested in different things, who in dressing, who on sports, who on improving the awareness about the quality of the technologies around us.
Who is contributing to open source often is part of the last group, maybe is someone curious because it's a user or because he found something that involves also the other interest's areas.

For example, I started coding my first website at 16 years old about… Dragon Ball, so pursuing my focus moved to develop games about it…

All this kinds of people need to be always motivated to do more or better to move on the focus of the project.
Also, you sometimes need a break or to be engaged again, it's a continuous work to stimulate them that someone should do in the right way.
People need to be gratified for what they do, in different ways and this is on you.
The coaching way of doing things is very helpful on motivating but also on optimizing a lot of common issues or tasks on project/people management. Coaching means that people around you succeed and it is something that you want in an OSS project.

What it means to be a mentor

The mentor is a person that follows you during your volunteers journey and needs to be ready for this task.
Also, his/her personal experience in the project or skills are very important. You are the first mentee of yourself because you approached probably the same issues, and you probably resolved them so you can give help quickly and easily.
It happens that people don't want to be involved as mentors because is time consuming, requires a lot of patience and a good mindset in communication and skills too.

This doesn't mean that you need all them but for sure being skilled in communication is important, because people will ask you a lot of things and on the way you reply, they will define how much they can trust in you and what they can ask you.
Who wants to talk to a rude person (and probably who likes pineapple pizza)? You will find someone that is more friendly, but this doesn't mean that you need to be a friend, only someone that you can talk to and that don't bites you.

A pragmatic analysis I get from the role mentor came from the article Why we stopped making Einsteins[196], where the reason basically is the fact that they had a tutor, or they got a personal teacher at home. This let them grow from the experience of skilled people that was shared to them when they were very young people.
A mentor is sharing own experience to others to let them grow with that like this book, just because we want to improve the world itself.

The Mentor's path:

- Analyze what kind of activities require a mentor and why
  - Verify the priority
- Be a friend/supporter of the mentee
- Understand the skills of the mentee
  - Help in case of missing skills
- Help people in doing activities that generate experience
- Be frank, constructive and work together for the value of the mentee itself
- Define your expectations
- Lead by example

There are two approaches for a mentor that are Formal or Informal and this depends on the type of activity.
The best way to be a mentor is not Authoritative or Democratic but in a Coaching way that is the best for educational and quality of the friendship in this task.
The coaching way is very simple and requires skills for specific activities as already explained in the document.

Don't forget that the relationship with a mentor is a long journey and requires a full view of the activities and goals.

You will fight between two aspects that are Convention vs Consistency, and they are both important. So you need to balance that in your routine!

Conventions are your workflow or the community convention on tasks, that you need to put in the plan but as mentor you need also to be elastic on managing the Consistency. Because you are helping a new volunteer that doesn't understand them probably, so initially you need to see which rules you can ignore and later "migrate" everything with the mentee.

[196] https://erikhoel.substack.com/p/why-we-stopped-making-einsteins?s=r

Motivate your mentees

The volunteers need to be motivated in their activities so don't forget to ask feedback on the activityitself about:

- What works? What doesn't?
- What would you like to do?
- Do you think these activities helped you to learn something? What?
- Do you think the workflow can be improved?

Remember feedback are important to gather information but also to give trust to the volunteer to be a part of a community and not a single and alone piece of a puzzle.

The rules are:

- Don't be defensive or against feedback but document all them, because they can help in a manner that you now don't imagine, in the future.
- each out often to the mentee or you will lose the volunteer, so get in touch twice a month it's the best timing initially (to not become annoying).

Remember feedback are not useful to improve only the activity/project but also yourself, to learn how to ask the right question in the right moment.
Every feedback needs to be explained to understand what is going on, yes or no are not enough.
Basically without context everything can be misconfused from everyone.

Also, you have to ask feedback to the volunteer to understand what's going on:

- Do you like to work on this project?
- Do you like to work in this team?
- What you want to do next?
- What was the main or bigger problem in the last months during activities?
- What are your activities like when you are proud of yourself?
- Did you learn something new?
- Did you need it a mentoring in the last months?
- Did you help someone as mentor?

The mentoring period depends on your needs but the ideal is one month because the people can remember better what they have done.
Also, trust the feedback from people with experience but also the feedback from new people because they have a different point of view.

When a new product is in alpha/beta status is not good to send to review to all the influencers or investors. Instead it's sent to a group of testers to validate the assumptions, and this can be done also inside the community. This can help to draft a better activity or roadmap that later can be disclosed or discussed with all the community. This helps on avoiding forgetting something and checking that the testers are aligned and can understand everything. What will happen when you disclose something that wasn't analyzed in the right way? Noise, discussions, ranting, rumors and so on. Open source doesn't need them so often even if everyone loves *flame wars*.

How to fight demotivation

Demotivation happens usually for 3 reasons:

- Lacking confidence on the project/activity
  - You need to understand the reason and create confidence
- Poor results
  - Define new objects and understand what was the reasons
- Hints not enough
  - Ask yourself what it's missing and do it

The conflicts also happen for the same reasons but also for:

- Missing information
- Goals not defined

To improve and avoid these problems usually you can start to improve the communications because often the misunderstanding is there.