Long running OpenSource projects

...

Survival guide

Topics

- The Open Source economy
  - Making money for free
- Surviving technical debt
  - Choosing and maintaining a technology stack
- The art of compromises
  - Building a Community around your project

The Open Source economy

Photo credit: (c) O'Reilly - Oscon 2002

When we call software "free," we mean that it respects the users' essential freedoms: the freedom to run it, to study and change it, and to redistribute copies with or without changes. This is a matter of freedom, not price, so think of "free speech," not "free beer."

The Open Source economy

Photo credit: (c) O'Reilly - Oscon 2002

Collaboration is an emerging pattern, and firms approaching software production in terms of sharing and cooperation are Free Software's good friend. Commercial Open Source, as far as based on participation and fostering communities, is aimed at promoting just the same idea of freedom, no less.

The Open Source economy

Let's focus on Open Source

Too many people misunderstand the meaning of free

in free software

And translate it into "gratis"

The Open Source economy

Sharing source code is a win - win approach

You get "free" code that you can include into your applications

(ReactJS, AngularJS and most of the libraries you use every day are all open source software)

The Open Source economy

Sharing source code is a win - win approach

You contribute your fixes / improvements to projects and libraries you use

The Open Source economy

Many major companies understood that

Nowadays we have a lot of Open Source tools too

(Visual Studio Code is Open Source)

The Open Source economy

Advantages as a Developer

- You can share your work and get feedbacks (for free)
- You can learn from other's work
- You can do self-marketing by creating open source projects or collaborating with existing ones
    - This will improve your CV with something that can be verified by employers

The Open Source economy

Advantages as a Company

- You can fix bugs in the code you use without having to pay someone else for that!
  - Or you can pay, you have a choice (with closed-source software you DON'T)
- You can get tailor made libraries and software
  - Through collaboration or funding you can drive other's software planning and get exactly what you need
- You can take advantage of other people's contributions to your software
  - Functionalities, bug fixing, translations, documentation, etc.

The Open Source economy

Common Myths

If I make my code public anyone can steal my ideas

Forget about it: if someone just clones your work, without adding any value to it, he won't be able to sell it

If anyone creates good software using your work, he is just creating something new without reinventing the wheel

The Open Source economy

Open Source Utopia

"Do ut des"

So that every involved entity

And the whole society

Can improve

The Open Source economy

You can build a fence, pay for it, and your neighbor will have a fence too, without paying any money.

Or you can share the cost and effort with your neighbor, so that both will have a fence and both will pay less for it.

But you can also avoid to build a fence, and both will have a bigger garden to share.

The Open Source economy

How can I make money then?

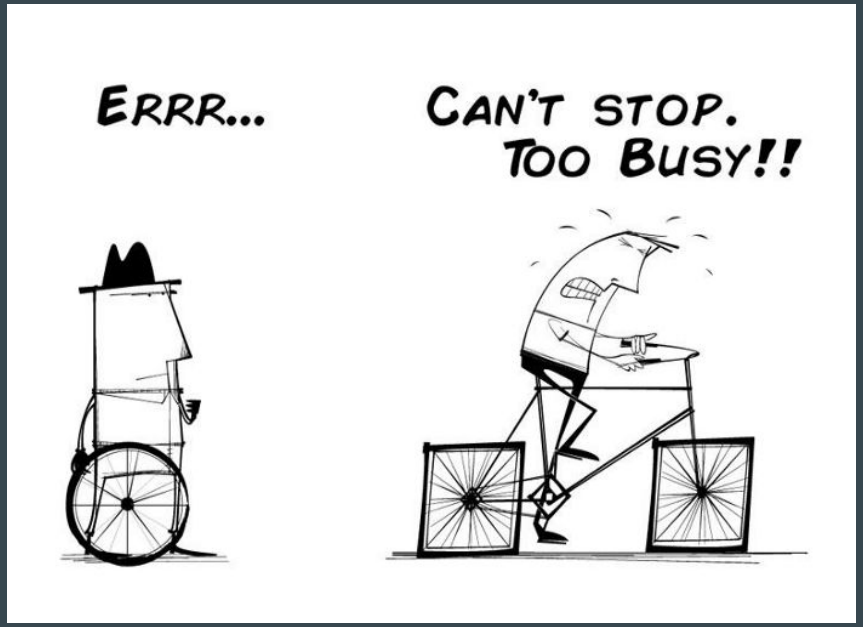
Creating valuable services using Open Source software

Creating tailor made software for your customers

Trainings and support

Fundings for bug fixing and improvements

Surviving Technical Debt



When you work on a long running project, choices are important, because what you decide now, may impact you for years

Surviving Technical Debt

Try to apply good practices from the beginning

Unit tests

Documentation

Styling guidelines

CI tools

...

Surviving Technical Debt

Time is limited:

Do most important things first!

Most important things are those

That save you time later!

Automate everything you can!

Surviving Technical Debt

Time is limited:

Automate running tests

Automate style guide checks

Automate deploys

Automate documentation building

Surviving Technical Debt

Time is limited:

Choose what to document first!

It depends on your target:

If you are building a framework write developers documentation:

Tutorials and how-tos first,

Reference after (and use proper tools)

Surviving Technical Debt

Time is limited:

Choose what to document first!

It depends on your target:

If you are building a tool / product for final users

Build users documentation first

(but you are a developer, so delegate this to someone else)

Surviving Technical Debt

Time is limited:

Choose what to document first!

It depends on your target:

If you want to do both...

Surviving Technical Debt

Time is limited:

Choose what to document first!

It depends on your target:

If you want to do both... you are in trouble!

(but that's another story)

Surviving Technical Debt

Limit the # of things to learn

New developers will want to work on your project

(hopefully)

and be productive as soon as possible

So ... make their life (and yours) easier!

Surviving Technical Debt

Limit the # of things to learn

The (unsolved) problem

Of code readability

"Readable" is programmer for "written by me"

(@acdlite)

Surviving Technical Debt

Limit the # of things to learn

The (unsolved) problem

Of code readability

Code can be poetry... and need (a lot of) comments

To be understood

Surviving Technical Debt

Limit the # of things to learn

The (unsolved) problem

Of code readability

Code can be mathematics... and need theorems demonstrations to be explained

Surviving Technical Debt

Limit the # of things to learn

The (unsolved) problem

Of code readability

Code can be a crime novel... where at the end you understand who is the murderer!

Surviving Technical Debt

Limit the # of technologies / libraries you use

Don't try to always include the latest / fanciest tech

(use a side project for that)

Same as above, less things to learn

Less things to maintain and upgrade

Surviving Technical Debt

If you are lucky

you will be able to upgrade to the latest releases

from time to time

Assuming new releases don't break too many things

If not you will stick to current releases

(almost) forever

Surviving Technical Debt

As long as you can

Don't do full (or almost full) rewrites

If you have to do

(we had)

Try to do something different

Instead of rewriting the same thing

The art of compromises

Working in the open means you have to interact everyday with a community of different people, and the way you do it can be fundamental to decide between success and failure for your project

The art of compromises

Building a community is a tough work

You have to help people for the sake of it

Maybe you will be thanked from time to time

Maybe you will struggle to understand what a user

is trying to do

The art of compromises

Users are those kind of people

Trying to use your creation

In weird ways

And pretending it should work

The way they think it should work!

The art of compromises

As developers, this is how we view end users

The art of compromises

Sometimes you will have some satisfaction

Such as a user you helped, becoming

a paying customer

Sometimes you will only get frustrated

But in the long term this kind of work pays

The art of compromises

As a developer:

- You get feedbacks from real users
  - Not only from your colleagues that are not that weird!
- You can get "famous", in your community
  - And use this popularity to enrich your CV
- You can get allies outside of your company to accomplish your owns targets
  - That maybe your company doesn't know they should care about

The art of compromises

As a company:

- You have a direct, informal, channel to potential customers
  - Where you can show what you can do
- You can share ideas with a wider audience
  - And get ideas directly from your potential users
- You can find other companies to work on a common target
  - And let your project grow

The art of compromises

Rules for a peaceful living in an open world:

- Contribute what you can
- Don't be too shy, share your ideas
- Be polite, give suggestions instead of expressing desires
- Help, don't ask
- When you take, say thanks (better if by a meaningful contribution)
- Do not fork... unless it's a matter of life and death... and then... do not fork (joking, but at least do it as the last option)

The art of compromises

And please, do not ask this:

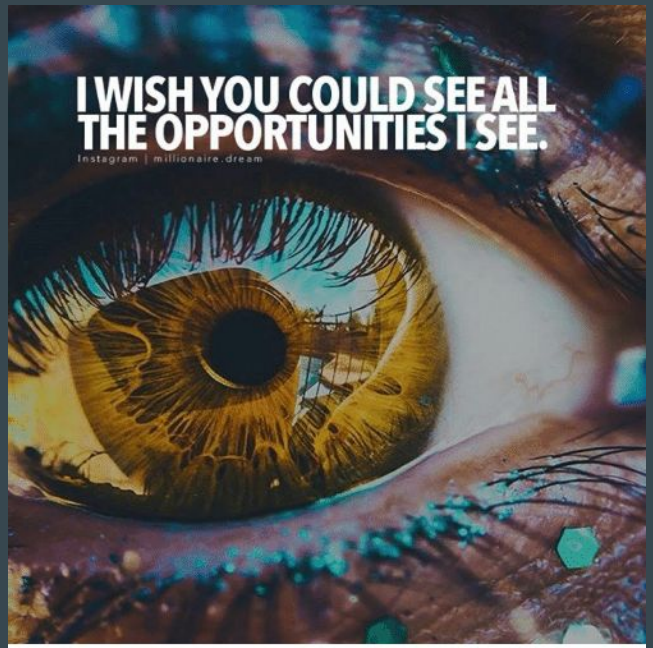
"URGENT!!! I need this to be fixed yesterday or I won't use your software anymore!"

Because... well... we will be sad (really?) to see you go away... but...

Time is limited, and we want to spend it for people who deserve it!

You always have a choice in the open source world!

The END



Opportunities are everywhere!

Thanks for listening!

maurobartolomeoli@gmail.com

@mauro_bart