Which factors determine the success of an open source project? [closed]

Asked 16 years, 3 months ago Modified 9 years, 6 months ago

Viewed 658 times

6

Closed. This question needs to be more focused. It is not currently accepting answers.

Want to improve this question? Update the question so it focuses on one problem only by editing this post.

Closed 10 years ago.

Improve this question

We have a series of closed source applications and libraries, for which we think it would make sense opening up the source code.

What has been blocking us, so far, is the effort needed to clean up the code base and documenting the source before opening up.

We want to open up the source only if we have a reasonable chance of the projects being successful -- i.e. having contributors. We are convinced that the code would be interesting to a large base of developers.

Which factors, excluding the "interestingness" and "usefulness" of the project, **determine the success of an open source project?**

Examples:

- Cleanliness of code
- Availability of source code comments
- Fully or partially documented APIs
- Choice of license (GPL vs. LGPL vs. BSD, etc...)
- Choice of public repository
- Investment in a public website

language-agnostic open-source project-management

Share

Improve this question

Follow

edited Jun 24, 2015 at 16:35

durron597

32.3k ● 18 ● 102 ● 160

asked Sep 17, 2008 at 8:05

Sklivvz

31.1k ● 24 ● 118 ● 174

10 Answers

Sorted by: Highest score (default)

There are a several things which dominate the successfulness of code. All of these must be achieved for

5

the slightest chance of adoption.

- Market - There must be a market for your open source project. If your project is a orange juicer in space, I doubt that you'll be very successful. You must make sure your project gets a large adoption amongst users and developers. It is twice as likely to succeed if you can get other corporations to adopt it as well.
- Documentation - As you touched on earlier documentation is key. Amongst this documentation is commented code, architectural decisions, and API notes. Even if your documentation contains bugs, or bugs about your software it is ok. Remember, transparency is key.
- Freedom - You must allow your code to be "free" - by this I mean free as in speech, not as in beer. If you have a feeling your market is being a library for other corporations a BSD license is optimal. If your piece of software is going to run on desktops then GPL is your choice.
- Transparency - You must write software in a transparent environment. Once you go open source there is no hidden secrets. You must explain everything you do, and what you are doing. This will piss off developers like no other
- Developer Community - A strong developer community is required. This must be existing. Only about 5% of users contribute back to the project. If

someone notices there haven't been any releases for a year they wont think "Wow, this piece of software is done," they will think "developers must of dumped it." Keep your developers working on it, even if it means they are costing you money.

- Communications - You must make sure you community is able to communicate. They must be able to file bugs, discuss workarounds, and publish patches. Without feedback, it is pointless to opensource the project
- Availability - Making your code easy to get is necessary, even if it means pissing off lawyers. You have to make sure your project is easy to download, and utilize. You don't want the user to have to jump through 18 nag screens and sign a contract in order to do this. You have to make things simple, and clean

edited Sep 17, 2008 at 8:22

answered Sep 17, 2008 at 8:16

Sargun Dhillon
1,848 ● 2 ● 19 ● 24

I think that the single most important factor is the number of users that are using your project. Otherwise its just a really well written, usefull and well documented bunch of stuff that sits on a server not doing very much...

Share Improve this answer

Follow

answered Sep 17, 2008 at 8:26

Jan Hančič

53.9k ● 17 ● 98 ● 101

To acquire contributors, you first need users, then you need some incompleteness. You need to trigger the "This is cool, but I really wish it had this or was different in this way." If you are missing an obvious feature, it's extremely likely a user will become a contributor to add it.

Share Improve this answer

Follow

answered Sep 17, 2008 at 8:27

jwanagel

4,059 ● 2 ● 27 ● 33

The most important thing is that the program be good. If its not good, nobody will use it. You cannot hope that the chicken-and-egg will reverse and that people will take it for granted until it becomes good.

Of course, "good" merely means "better than any other practical option for a significant number of people," it doesn't mean that its strictly the best, only that it has some features that make it, for many people, better than other options. Sometimes the program *has* no equivalent

anywhere else, in which case there's almost no requirement in this regard.

When a program is good, people will use it. Obviously, it has to have a market among users--a good program that does something nobody wants isn't really good no matter how well its designed. One could make a point about marketing, but truly good products, up to a point, have a tendency to market themselves. Its much harder to promote something that isn't good, so clearly one's first priority should be the product itself, not promoting the product.

The real question then is--how do you make it good? And the answer to that is a dedicated, skilled development team. One person can rarely create a good product on their own; even if they're far better than the other developers, multiple perspectives has an incredibly useful effect on the project. This is why having corporate sponsors is so useful--it puts other developers' (from the corporation) minds on the problem to give their own opinion. This is especially useful in the case that developing the program requires significant expertise that isn't commonly available in the community.

Of course, I'm saying this all from experience. I'm one of the main developers on x264 (currently the most active one), one of the most popular video encoders. We have two main developers, various minor developers in the community that contribute patches, and corporate sponsorship from Joost (Gabriel Bouvigne, who maintains

ratecontrol algorithms), from Avail Media (who I work for sometimes on contract and who are currently hiring coders on contract to add MBAFF interlacing support), and from a few others that pop up from time to time.

One good developer doesn't make a project--many good developers do. And the end result of this is a program that encodes video faster and at a far better quality than most commercial competitors, hardware or software, even those with utterly enormous development budgets.

Share Improve this answer edited Sep 17, 2008 at 8:50

Follow

answered Sep 17, 2008 at 8:34

Dark Shikari

8,009 ● 4 ● 28 ● 38

1

In looking at these issues you might be interested in checking out the online version of a course on open source at UC Berkeley, called Open Source Development and Distribution of Digital Information: Technical, Economic, Social, and Legal Perspectives. It's co-taught by Mitch Kapur (Lotus founder) and Paula Samuelson, a law school professor. I have a long commute and put the audio of the course on my iPod last year – they talk a lot about what works, what doesn't and why, from a very broad (though obviously academic) perspective.

Share Improve this answer answered Sep 25, 2008 at 22:17

Follow

Will M
2,491 • 2 • 18 • 19

1

Books have been written on the subject. In fact, you can find a free book here: producing open source software

Share Improve this answer

Follow

answered Sep 25, 2008 at 22:20

Jason Baker
198k • 138 • 382 • 520

0

Really, I think the answer is 'how you run the project'.

All of your examples matter, yes, but the key things are how the interaction between developers is managed, how patches etc are handled/accepted, who's 'in charge' and how they handle that responsibility, and so on and so forth.

Compare and contrast (the history isn't hard to track down!) the management of the development of Class::DBI and DBIx::Class in Perl.

Share Improve this answer

Follow

answered Sep 17, 2008 at 8:12

Penfold
2,568 • 16 • 16

0

I was just reading tonight an excellent post on the usability aspect of successful vs unsuccessful open source projects.

Excerpt:

> A lot of bandwidth has been wasted arguing over the lack of usability in open-source software/free software (henceforth “OSS”). The debate continues at this moment on blogs, forums, and Slashdot comment threads. Some people say that bad usability is endemic to the entire OSS world, while others say that OSS usability is great but that the real problem is the closed-minded users who expect every program to clone Microsoft. Some people contend that UI problems are temporary growing pains, while others say that the OSS development model systematically produces bad UI. Some people even argue that the GPL indirectly rewards software that’s difficult to use! (For the record, I disagree.)
>
> http://humanized.com/weblog/2007/10/05/make_oss_humane/

Share Improve this answer

Follow

answered Sep 17, 2008 at 8:13

micahwittman

12.5k ● 2 ● 34 ● 37

Just open-source it. Most probably, nobody will start contributing yet. But at least you can write on the press-releases that your product is GPL or whatever.

The first step is that people start using it...
And maybe then, after users get comfortable, they will start contributing.

0

Share Improve this answer

Follow

answered Sep 17, 2008 at 9:00

user11104

382 ● 1 ● 3 ● 13

Everyone's answers have been good so far, but there's one thing missing and that's good oversight. Nothing kills an open source project faster than not having some sort of project management. Not to tell people what to do so much as to just add some structure and tasking for the developers you are hoping to attract.

Disorganized projects fall apart fast. It's not a bird you just let go and watch it fly away.

0

Share Improve this answer

Follow

answered Sep 23, 2008 at 14:08

MattC

12.3k ● 10 ● 56 ● 78