(/)

A Guide To Participating In An Open Source Software Community

by **Stuart Yeates** on 1 June 2005 , last updated 9 September 2013

In this video we discuss enganging in open source communities.

Participating in an open source software community can initially seem an intimidating prospect. However, such communities are ultimately composed of people, with all the virtues and foibles of people everywhere. Preconceptions should be avoided. Open source software communities are rarely populated entirely by highly technical individuals proud to call themselves *hackers*. Nor are they composed entirely of certified software engineers. You will get along better by simply engaging with the community as you find it, leaving any preconceptions at the door.

Remember, useful skills and a bit of effort will always be welcome. Here are some tips that may make your progress a little easier.

Prepare

Play to your strengths

Not everyone has the same skills, and there are many roles outside writing code in open source projects. These include:

- documentation writers
- translators
- website designers
- GUI designers, and
- communications people

Projects also need people to maintain the build and test machines, people to support new users, and beta-testers. See our separate document, Roles in open source projects (rolesinopensource), for more details.

Estimate your time commitment

Decide whether you want (and are able) to commit a small amount of time every day or a more substantial amount of time periodically. Be careful to offer to take on only as much as you can realistically deliver; people respect completed work far more than hollow offers of help. Think carefully about where the time for your efforts will come from. If you are working on the project as part of paid work, as most people do, you’ll need to justify that time to your boss. On the other hand, if you are spending some personal time on a project, be sure to allow time for your family and friends.

Check your employment contract

If you intend to work on a project as part of your day job, it is vital to check your employment contract for intellectual property rights (IPR) clauses to ensure that you are permitted to participate (contributing) in an open source project. Since software code is copyright protected, only the owner of that code - or someone authorised by the owner - may legally license it for others to use. Many employers recognize the value of staff participating in the development of software that is used within the firm. Some employers have taken the next step and mark this recognition within institutional IPR policies and/or employment contracts. If you are an educator planning to involve students in an open source project, you should also check that their student agreement permits this type of involvement and that the students understand the implications.

Get To Know Your Community

Understand how the community communicates

Most communities have an accepted way of engaging with the community. This can be as simple as joining a developers' email list, and/or adopting a particular code of practice. Most important, perhaps, is to learn how questions are asked and answered in this particular development community. The paper How To Ask Questions The Smart Way (http://catb.org/~esr/faqs/smart-questions.html) by Eric Raymond and Rick Moen is an excellent guide. It is worth spending some time getting to know your community before you participate in it.

Understand how the community is governed

Some communities, for example, that of the Linux kernel, are hierarchies with clear chains of command. Others, such as the community around the Debian distribution, are flat democracies. These different types of communities require different styles of participation (governanceModels). Understanding how decisions are made and conflicts resolved will help you understand how to best engage with the community.

Understand the role of constructive criticism

The culture in a particular open source community may vary from that in other communities. Just as different organisations operate with different cultures, there are differences in how these communities communicate. Discussions can be lively, in which a range of individuals frankly express their opinions. The end goal is to further the development of the software and community as well as possible; this is certainly true for mature communities that have proven to be successful. Critisism is in general always directed towards that goal, and not to the person as an individual. An exception to this are the 'poisonous people', a phenomenon that is further described below.

Get to know the people

Large open source projects have a wide range of participants and the first or loudest answer to a query is not always the correct answer. By following the project for a short time you can gain useful knowledge of the people involved and their roles. Not all open source communities are wholly virtual. Many of the big projects have face-to-face get togethers to discuss future plans, or run code fests, and these events can be used to establish personal relationships which will aid online discussion. When done well, these physical get togethers will always be reflected in online communications, such as tweets, slides posted online, and meeting minutes.

Understand the communications channels

Open source projects typically use chat sessions (such as IRC or jabber), mailing lists, websites, blogs, wikis, and version control repositories as their primary means of communication. Get to know your project's communication style and preferred tools (communitytools).

Learn about 'Poisonous People'

Some behaviours in open source communities are seen as anti-patterns[1] in that they obstruct constructive activity. Most people will inadvertently engage in one or more of these activities from time to time, usually with good intention. We strongly recommend you spend 55 minutes watching 'How To Protect Your Open Source Project From Poisonous People' (http://www.youtube.com/watch?v=ZSFDm3UYkeE) by Ben Collins-Sussman and Brain W. Fitzpatrick. This video will not only show you which activities are not constructive, but also help you understand how to limit their impact when you see them in others.

Engage

Communicate what you are working on
If you work completely on your own to re-develop part of a project, by the time you finish, someone will almost certainly have either duplicated your effort or the project may have made other changes that render your work obsolete. A corollary of the lack of central planning and coordination in open source projects is a need for everyone to bear at least a portion of the burden of communication.
Acknowledge resources you use and their creators
It is important to acknowledge the resources you use. This increases the likelihood that others will also find the resource, and provides positive feedback to the creators, encouraging them to maintain existing resources and develop new ones.
Give back
A healthy community depends on the principle of individuals giving back to that community. So, if you have received support from the community, the best way to make sure that situation continues is to reciprocate by providing support to another community member when you can.
Plan an exit strategy
Have a plan for what happens to your contributions to the open source project when your commitment to the project changes. If you are a educator planning to involve students in an open source project, have a plan for the end of the course, when the students move on. Bug fixes, well-integrated modules, additional translations, generic documentation and publicity material tend to survive when the creators leave the community. Local hardware, radical restructuring, and poorly integrated modules tend not to.
Retire gracefully
If your open source participation is waning due to work, family or other commitments, inform other community members of the problem, so that they can take up the slack and/or take steps to lower your workload.

Summary

For most people, their first foray into an open source project can be both difficult and scary. However, most quickly realise that it is well worth the effort. The rewards are higher quality software, reduced costs of development and an opportunity to learn from the best.

Finding a starting point can often be quite hard, but learning about the structure of a sustainable community project (howtobuildcommunity) is a useful step towards understanding how one can contribute to a community.

Further Reading

Links:

- ‘How To Ask Questions The Smart Way’ by Eric Raymond and Rick Moen [http://www.catb.org/~esr/faqs/smart-questions.html (http://www.catb.org/~esr/faqs/smart-questions.html)]
- How to Participate in the Linux Community [http://ldn.linuxfoundation.org/book/how-participate-linux-community (http://ldn.linuxfoundation.org/book/how-participate-linux-community)]
- How To Protect Your Open Source Project From Poisonous People [http://www.youtube.com/watch?v=ZSFDm3UYkeE (http://www.youtube.com/watch?v=ZSFDm3UYkeE)]

Related information from OSS Watch:

- Roles in open source projects (rolesinopensource)
- How to build an open source community (howtobuildcommunity)
- Contributor Licence Agreements (cla)
- Essential tools for running a community-led project (communitytools)
- Governance models (governanceModels)
- Avoiding abandon-ware: getting to grips with the open development method (odm)

---

1. http://en.wikipedia.org/wiki/Anti-pattern (http://en.wikipedia.org/wiki/Anti-pattern) ↩

Jump to section...

INFORMATION FOR

Newcomers (/Resources/Beginners)

Academic End Users (/Resources/Staffandstudents)

Software Developers (/Resources/Developers)

IT Managers And Technical Staff (/Resources/Managers)

Strategic IT Decision Makers (/Resources/Strategists)

RESOURCES

Strategy And Policy (/Resources/Stratpol)

Open Source Software Development (/Resources/Softwaredevelopment)

Intellectual Property Rights (IPR), Licensing And Patents (/Resources/Ipr)

Building Communities (/Resources/Buildingcommunities)

Case Studies (/Resources/Casestudies)

OSS Watch values your input and questions. If you have feedback on this document, or any OSS Watch activity, please send it to: info@oss-watch.ac.uk.