How to contribute to Open Source

Audris Mockus

CS340

Motivation

- Online activities became a de-facto resume for software developers
- Open source projects became critical for *ALL*
  - commercial, governmental, and non-governmental organizations
- Essential to understand and practice open source contributions
  - ensure that the actual contribution will succeed and,
  - you have good experiences while doing it.

How to choose an appropriate open source project to contribute?

- Start by thinking about the projects you have already used
- Choose the project you are very interested in
- Choose the project you are familiar with
- Choose the project that can be run stand-alone
- Choose an active project
- Check if the project is accepting contributions

Interest and familiarity

- Choose the project you are very interested in.
  - For example, if you are interested in search engine and want to know how it works. First, you should study technical literature to learn the basic technical points. After that, you can choose Lucene, Sphinx to learn through practice.
- Choose the project you are familiar with.
  - Are you familiar with the programing language and the technology?

Standalone and active

- Standalone
  - Most projects, are components/plugins/libraries, that would require you first to understand the underlying frameworks.
  - Some of the projects may require very complex installation and configure in order to test the software.

- Active
  - Number of participants
  - The frequency of the commits and issues
  - The enthusiasm of the discussion in the community (e.g. answers on SE)

Suitability Checklist I

- Meets the definition of open source
  - Does it have a license? Usually, this is a file called LICENSE in the root of the repository.
- Project actively accepts contributions
  - Look at the commit activity on the master branch. On GitHub, you can see this information on a repository’s homepage.
  - When was the latest commit?
  - How many contributors does the project have?
  - How often do people commit? (On GitHub, you can find this by clicking "Commits" in the top bar.)

Suitability Checklist II

- Look at the project’s issues.
  - How many open issues are there?
  - Do maintainers respond quickly to issues when they are opened?
  - Is there active discussion on the issues?
  - Are the issues recent?
  - Are issues getting closed? (On GitHub, click the "closed" tab on the Issues page to see closed issues.)

Suitability Checklist III

- Look at project’s pull requests.
  - How many open pull requests are there?
  - Do maintainers respond quickly to pull requests when they are opened?
  - Is there active discussion on the pull requests?
  - Are the pull requests recent?
  - How recently were any pull requests merged? (On GitHub, click the "closed" tab on the Pull Requests page to see closed PRs.)

Suitability Checklist IV

- Is the project welcoming?
  - A project that is friendly and welcoming signals that they will be receptive to new contributors.
  - Do the maintainers respond helpfully to questions in issues?
  - Are people friendly in the issues, discussion forum, and chat (for example, IRC or Slack)?
  - Do pull requests get reviewed?
  - Do maintainers thank people for their contributions?

Understand how the project works

- Inspect main documents:
  - documents may be more important than reading code at first
  - Find in the top level of a repository,
    - license, readme, contributing, tutorials and so on.
  - E.g. README is the instruction manual that
    - welcomes new community members to the project.
    - explains why the project is useful and how to get started.

- E.g., In Linux Kernel community
  - How to do Linux Kernel development
    - contains some rules of the community
    - tells contributors what they should pay attention to

How contributors collaborate?

- You should understand the way that contributors collaborate
  - Issue tracker: Where people discuss issues related to the project. Maybe you could find the bug there you want to deal with.
  - Pull requests: Where people discuss and review changes that are in progress.
  - Discussion forums or mailing lists: Some projects may use these channels for conversational topics. Others use the issue tracker for all conversations.
  - Many contributors’ first interactions with the Open Source community usually happen there. Please subscribe the project’s mailing list, thus you can receive interesting discussions and sharing your idea.
- If the project is on GitHub, check out Make a Pull Request, which @kentcdodds created as a walkthrough video tutorial.
- Practice making a pull request in the First Contributions repository, created by @Roshanjossey.

How to make your first contribution?

- 1. Send an email
    - The first contribution usually is not writing code.
    - Researchers found that often a significant period of observation (lurking), ranging from a couple of weeks to several months, was needed before a joiner became a developer [2].
    - However, the standard deviation was quite high, due to the variance of technical skills and the time that the contributors devote.
    - During this time, new contributors usually participate in discussion via email, I.e., join the mailing list.

What to use the first contact for?

- 1) Ask question
- 2) General technical discussion
- 3) Usage feedback (no bug report)
- 4) Request for help
- 5) Point to technical resources/ refer to other projects
- 6) Express interest to contribute
- 7) Suggestion for improvements
- 8) Propose/outline bugfix (no code)
- 9) Coordination and organization discussion
- 10) User support
- 11) Answer (technical) question
- 12) Self introduction
- 13) Announcing “external” contribution
- 14) Give Feedback on others contribution

Use issue tracker

- Report an issue
  - Before reporting an issue, you need make sure this issue has not been reported by others (You could check it in the issue tracker system).
  - Then, you need to describe what the issue is, the context and how to reproduce it.
- Comment on an issue
  - These comments might be an interpretation for a confusing report or suggesting a possible solution.
  - Researchers [3] found that starting from a comment instead of reporting an issue also reflects intention of getting involved, perhaps by first finding a similar issue and commenting on it instead of simply reporting an issue encountered as a user.

Commit code (create a pull request)

- It is much difficult for new contributors to contribute code. If you want to challenge yourself at beginning, the first thing you should do is choosing the appropriate component. The following three items may help you [4].
  - Choose the component which is easy to modify and coding.
  - Some components perform simple tasks, such as documentation
  - Others may be more difficult than others: try to avoid these kind of components.
- Choose the component which has less dependent on others.
  - This allows contributors to use existing functionalities without having to understand the rest of the specific functionalities used by other components. And such components can be added or removed at any point, often without having to recompile the whole source code.

You submitted the contribution: Now What?

- No respose in a week
  - politely ask for a review in that same thread, asking someone for a review, can @-mention if you know who is the relevant person
  - **Don't** reach out to that person privately; remember that public communication is vital to open source projects.
  - It's possible that nobody will respond, ever
    - Possibly personal circumstances
    - Try to find another project or way to contribute
- Someone requests changes to your contribution
  - common feedback on
    - the scope of your idea, or
    - changes to your code
  - be responsive -- opening a PR and walking away is **bad** If you don't know how to make changes, research the problem, then ask for help if you need it
  - if you don't have time to work on the issue anymore, let the maintainer know so they're not expecting a response
- Your contribution doesn't get accepted.
  - If you're not sure why: ask the maintainer for feedback and clarification
  - Don't argue or get hostile. Fork and work on your own version if you disagree!
- Your contribution gets accepted

References

- [1] How to contribute to Open Source? https://opensource.guide/how-to-contribute/#orienting-yourself-to-a-new-project
- [2] Krogh G V, Spaeth S, Lakhani K R. Community, joining, and specialization in open source software innovation: a case study[J]. Social Science Electronic Publishing, 2003, 32(7):1217-1241.
- [3] Zhou M, Mockus A. What make long term contributors: willingness and opportunity in OSS community[C]// International Conference on Software Engineering. IEEE, 2012:518-528.
- [4] Xin Tan, Hanmin Qin, Minghui Zhou: Understanding the Variation of Software Development Tasks: a Qualitative Study. Internetware 2017: 15:1-15:6