Design research

Overview

1. OSS space
2. Existing roles
3. Typical OSS project lifecycle
4. Funding OSS today
5. Tools overview
6. Next steps

1. OSS space today

Problems

OSS space

- Lack of funding for project maintainers
- Weird status of OSS as commons: if you write code and keep it to yourself you have more chance of being able to make money off of it
- Volunteer burnout
- Community mismanagement

Problems

OSS space

- Sense of alienation: you only know a small part of your community, only the ones that tell you what's wrong.
- The infrastructure people rely on is actually not properly maintained

Opportunities

OSS space

- Adding monetary reward might prompt maintainers to take the built quality of their OSS more seriously
- Having access to primitives to create your own tooling
- Allow easier & more forms of contributions
- Providing more flexible & accessible funding options

Opportunities

OSS space

- Demistify licensing
  - Set some standards when it comes to licensing, protecting the casual contributors to not be exploited. (eg. Free to try, pay to use in commercial space)
- Visualise impact of OSS project to give contributors a sense of reward

2. Existing roles

Existing roles

1. Maintainer
2. Contributor
3. User

1. Maintainer

Description

Maintainer

Contributors who are responsible for driving the vision and managing the organizational aspects of the project. Maintainers are the most important actors within an open source project. They have often initiated the project.

Main responsibility

Maintainer

- Authoring: author git commits on the git history
  - Converging commits: merge commits or that incorporate other branches/forks
  - Extending commits: extending the history of the codebase (every other commit)

Main responsibility

Maintainer

- Request management: manage the backlog of requests
- Publishing & versioning: publish releases (distribution/naming/versioning)
- More detailed list here

Painpoints

Maintainer

- Lack of funding leads to frustration (My code is free, my time isn’t)
- OSS work is the most important in their career but they can’t get paid for it
- It’s harder to accept contributions then to make them
- Time management

Painpoints

Maintainer

- Hard to stay motivated after initial problem they set out to solve with the project is solved
- Very ingrateful job to be governing OSS projects, every users and contributor has their selfish reason to be there, governing is mostly about saying “no”
- Hard to find other maintainers/contributors to share the load with

Painpoints

Maintainer

- They are reluctant to start new OSS projects because of the work it entails
- Tools are catered to everybody, making them lack specificity for maintainers

Success criteria

Maintainer

— Monetize

— Getting recognition

— Having recurring contributors

— Their OSS project doesn’t need them anymore

Existing tools

Maintainer

— Version Control System

— Bug & Issue tracker

— Archiving & release management

— Proposals (PR's)

— Discussion forums or mailing lists

— Synchronous chat channel

— Tools for funding (Patreon + opencollective)

Job stories

Maintainer

When I maintain my open source projects I want:

- to be able to make a living so I can continue the work that I find meaningful.
- to see how many people use my project so I can get an idea of what impact I’m having.
- the tooling to be for free so that the entry barrier for contributors is as low as possible.

Job stories

Maintainer

When I maintain my open source projects I want:

— to have one streamlined workflow so I can work as fast and efficient as I can.

— to have a strong community around the project so I can delegate some of the work and decisions to them.

Job stories

Maintainer

When I switch over to a new tool I want to keep my existing workflow intact so I can switch over with minimal impact on my day to day.

2. Contributor

Description

Contributor

Could be anyone who comments on an issue or pull request, people who add value to the project (whether it’s triaging issues, writing code, or organizing events), or anybody with a merged pull request (perhaps the narrowest deﬁnition of a contributor).

Main responsibility

Contributor

- Authoring: author git commits on the git history
  - Extending commits: extending the history of the codebase (every other commit)
- Commenting & issue submission

Painpoints

Contributor

- Incomplete or confusing documentation
- Unresponsiveness of maintainers
- Dismissive responses, leading to contributors leaving
- Conflict through bad communication

Painpoints

Contributor

- Unexplained rejection making contributors feeling like they wasted their time
- Proposals not being accepted leading to contributors moving on

Success criteria

Contributor

- Original bug/problem is solved
- Merged contribution
- Getting recognition

Existing tools

Contributor

- Version Control System
- Bug & Issue tracker
- Proposals (PR's)
- Discussion forums or mailing lists
- Synchronous chat channel

Job stories

Contributor

When I contribute to OSS I want:

— to know where I can have most impact so I can just focus on programming.

— to see that an OSS project is maintained so I can contribute where it matters.

— one accessible & searchable place to see all the most relevant OSS projects so I can easily choose where to contribute.

3. User

Description

User

People who use the OSS in their project. They might be active in conversations or express their opinion on the project’s direction.

Main responsibility

User

- Use OSS in their work
- Report bugs

Painpoints

User

- Relying on OSS that is not actually maintained properly or doesn’t have a robust built
- Incomplete or confusing documentation
- Reported bugs don't get fixed and questions go unanswered

Success criteria

User

- Bugs get resolved quickly & often
- Project is continuously maintained
- Accessible and understandable licensing

Existing tools

User

- Version Control System
- Bug & Issue tracker
- Discussion forums or mailing lists
- Synchronous chat channel
- Source code scanning and license compliance (more corporate)

Job stories

User

When I look for open source projects I want to find projects that solve my problems so I can focus on only building what is necessary.

When asking questions on an OSS project I want to have them answered quickly so I can continue my work.

Job stories

User

When I want to use an OSS project:

- I want to know the project is actively being maintained so I can be sure the dependency is not gonna get stale.
- I want to have good documentation so I can see if the project solves the problem I set out to solve.
- I want them to have good/clear licenses so I am

3. Typical OSS project lifecycle

Phase 1: Concept

Typical OSS project lifecycle

One person is running into trouble trying to build something, is looking around for projects to solve their problem. They see other people are asking the same questions, but are unable to find a ready made solution for it.They decide to build the solution themselves, they start a new project on Github.

Phase 2: Bootstrap

Typical OSS project lifecycle

They are just tapping away, trying to solve this problem. Maybe they share it with a friend developer to get some extra eyes on it. Will go to some IRC channels asking some specific questions.

Phase 3: Early Development

Typical OSS project lifecycle

There are a few more people interested, your solution is basic but it works. The few conversations they have about this project is with the one or 2 befriended developers they have told about the project. They write a small readme more for themselves then anybody else.

Phase 4: Early Adoption

Typical OSS project lifecycle

After being online for a few months, some people have starred it on Github. They've even got a few issues that were submitted. Because they are so happy that a person found it useful, they solve the issue quickly. They add a basic contribution guide outlining what way they expect PR's or issues to be submitted.

Phase 5: Development

Typical OSS project lifecycle

About 50 people have starred the repo. A few more issues are coming in a week now, some people have started complaining about the lack of clear documentation. It quickly sinks in that the maintainer is going to have to spend more time on this. They continue to patch some stuff, still building mainly to solve their own problem.

Phase 6: Adoption

Typical OSS project lifecycle

More and more people are looking to contribute. The maintainer is starting to have certain issues that are a few weeks old. Also there are a few PR's that don't really reflect the direction that they want to take the project on.

Phase 7: Mainstream

Typical OSS project lifecycle

They stop really spending time on it since it solved the problem they set out to solve. They check in once every 3 months, ignoring the issues and conversations. Just goes straight to the PR's and seeing which ones have passed the tests, only those will they go through and merge in.

Phase 7: Mainstream

Typical OSS project lifecycle

This also shows the community that the repo is still being maintained. Nobody is ever really coming in trying to help out, so the project just lives on like this.

4. Funding OSS today

Bounties

Funding OSS today

- On a small scale, individuals or companies sometimes post “bounties”, which are cash prizes for certain development milestones.
- They can however create perverse incentives for contributing to a project

Services

Funding OSS today

- Consulting is a viable option for independent developers, if there are enough people using the project who are willing and able to pay for extra help.
- On a small scale, it can also distract developers from improving the project itself

Paid licenses:

Funding OSS today

— Some developers feel that licensing could provide at least a partial solution to open source’s funding problems. If open source projects are being heavily used, why not charge for them?

— This model is arguably at odds with the enormous social calie that open source has provided, which suggests that when software is free, innovation follows

Sponsorship or donations

Funding OSS today

- A project maintainer will look for benefactors who believe in the value of their work and are willing to financially support them. There are two major sources of funding at the moment: software companies and other developers.

Crowdfunding

Funding OSS today

— Similarly to bounties, crowdfunding can be useful for funding new features or development work with a clear, tangible outcome. It can also help reduce perverse incentives from bounties.

— Crowdfunding still does not address the need for funding related to ongoing operations and overhead, nor is it a recurring source of capital (time intensive)

Corporate sponsorship of infrastructure projects

Funding OSS today

- On a larger scale, in some instances, a project’s value becomes so widely regarded that a company will hire a contributor to work on the project full-time.
- A concern is that a single company could have undue influence over a project, since they are the only sponsor.

Administrative support and fiscal sponsorship

Funding OSS today

— Several foundations provide organizational support, such as fiscal sponsorship, to open source projects in other words, taking care of the non-coding tasks that many developers would prefer not to do.

— It is important to note that these foundations provide legal and administrative, but rarely financial, support.

Creating a foundation

Funding OSS today

— Creating a foundation is sensible for very large infrastructure projects, less so for smaller projects, due to the work resources and ongoing corporate sponsorship needed to create a sustainable organization.

— Qualifying as a 501 is challenging due to the lack of awareness about open source technology and to see OS as non-charitable activity.

Corporate programs

Funding OSS today

- Dedicating salaried time is one of the most important ways that companies contribute to open source
- Occasional donations to project crowdfunding campaigns
- “OS summer camps”: eg Google’s Summer of Code

Other institutional actors

Funding OSS today

- Support digital infrastructure in various ways
  - Github: chance to be open source utility
  - Venture Capital: have a stake in the future of digital infrastructure but they could not make investments into projects that didn’t have a business model
  - Academia: Play an important role in supporting digital infrastructure (new projects)

5. Tools

Tools

1. Git - Version Control System
2. GitHub / Gitlab / BitBucket / Phabricator
   - Git repo overview
   - Bug & Issue tracker
   - Archiving & release management
   - Proposals (PR's)

1. Git - Version Control System

The good

Git - Version Control System

- Very large adoption
- Accessible from CLI -> uninterrupted flow

The bad

Git - Version Control System

- Only part of the flow – Issues/proposals live somewhere else (no CLI)
- Steep learning curve for outsiders
- Multiple ways to use it, existing templates (Github-flow)

2. GitHub / Gitlab / BitBucket / Phabricator

The good

GitHub / Gitlab / BitBucket / Phabricator

— Focussed on collaboration

— Community is there

— Streamlining/unifying workflow

— Project discovery

The bad

GitHub / Gitlab / BitBucket / Phabricator

- Not opinionated in their target user -> for everybody
- Don't cover all the existing painpoints
- Missing important pieces like:
  - governance
  - monetization

Risk ending up with a list of features to build with no opinion

Missing context

- Missing shared product vision
- Missing main flows we want to work towards + prioritization
- Missing understanding around new roles we're adding - Token holder, validator

6. Next steps

Next steps

1. Collect ideas on how to generate shared vision
2. Translate that into a workshop
3. Run workshow & digest into a shared vision
4. Prioritize flows based shared vision
5. Result: Product strategy -> Why are we building what and when.