Day 2

Chan
Zuckerberg
Initiative

Open Source Citizenship

Participating in OSS as a corporate contributor

CZI SciTech Team Training | 31st August 2023

Introduction 👋

Goal: Understand how Open Source projects and communities work, and how to engage with them meaningfully.

Trainers: Tania Allard, Pavithra Eswaramoorthy

Questions: Ask any immediate questions in the chat, or at the end of each section when we open the room for discussion

This space is covered by the Contributor Covenant Code of Conduct.

Please report unacceptable behavior to opensource@chanzuckerberg.com.

Topics 🗂️

1. Introduction to Open Source
2. Participating in Open Source
3. Enhancing your OSS contribution impact

🔗 **Extended Training Material:**

https://github.com/chanzuckerberg/czi-oss-training

Outline: Day 2 🎯

- Participating in OSS [35 mins]
- Break [5mins]
- Enhancing your OSS contribution impact [20 mins]
- Hands-on activity [15 mins]

02. Participating in Open Source

Understanding Open Source Sustainability and Life Cycle

Sustainability is...

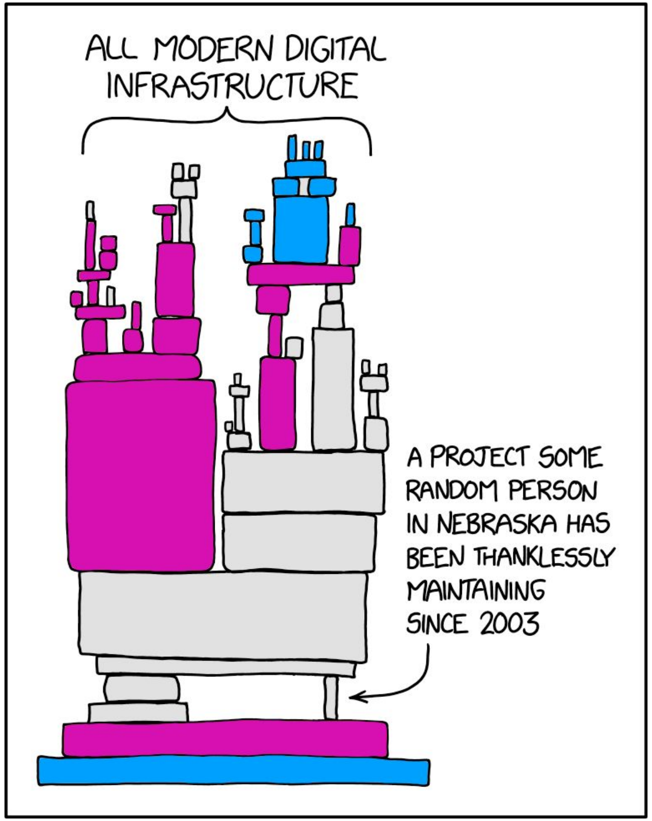
> “Development that meets the needs of the present without compromising the ability of future generations to meet their own needs.”
>
> *The Brundlant report*

Corporate Sustainability 101

A sustainability plan impacts **every aspect of an organization's operation** and how it engages with and impacts external groups.

⚠️ This applies to Open Source as well - funding is only but one aspect of sustainability

v2.0


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

🟦 Projects owned & resourced by multi-billion dollar company who hires full time engineers directly or through contracts

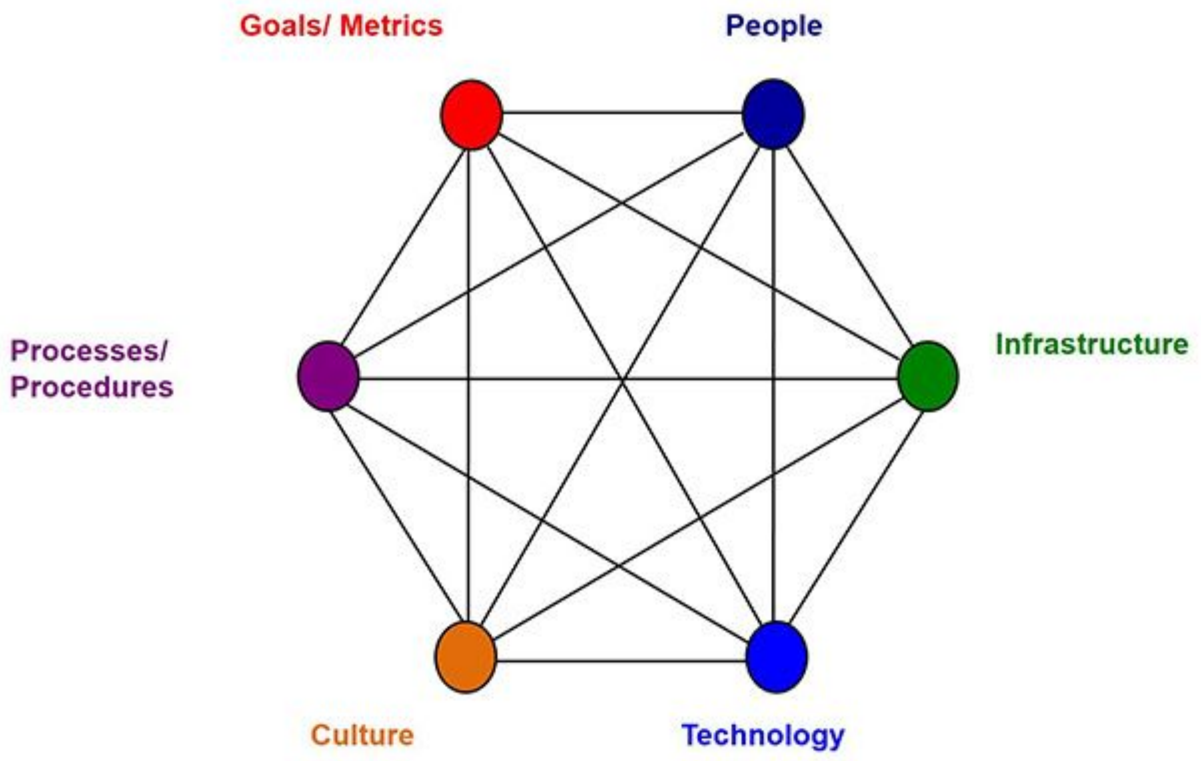
🟪 Projects governed by multi-billion dollar organization whose maintainers are not directly paid

⬜ Other

adapted from https://xkcd.com/2347/

Thar be dragons (PyCon AU 2023) - Amanda Casari

OSS and organizations are sociotechnical systems

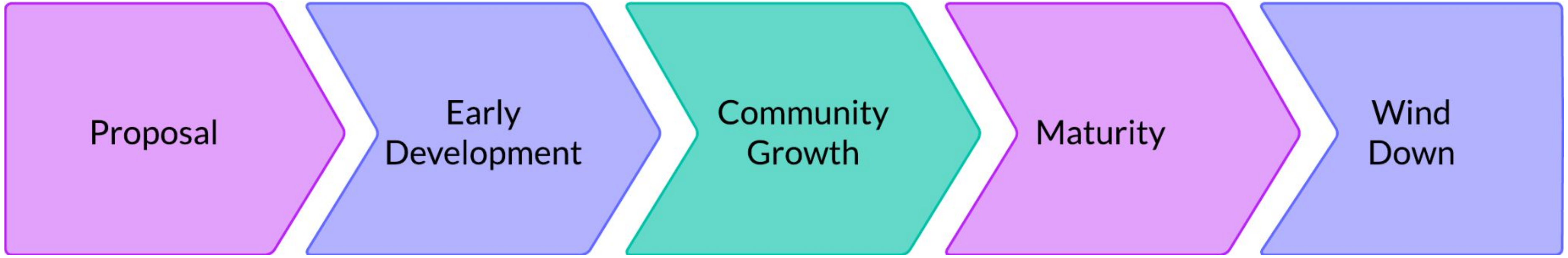


Integrating OSS in your Sustainability Plan 🌱

How do you decide where/how to contribute?

1. Identify the most critical OSS within your supply chain.
2. Identify the sustainability needs of those projects in collaboration with the community.
3. Determine how you can help meet those needs and keep engaging with the community.

OSS Project Life Cycle 🌲



Knowing where the project is can help identify the right level of support needed

🔗 Life Cycle deep dive in the course repository

1. Proposal

A problem is identified that new open source software can solve, and proposal is created with:

- Scope and goals for the project
- User workflows and comparisons to alternatives
- Software development stack and system architecture
- Development timeline and deliverables
- Resource requirements (contributors and funds)

Teams look for incubation opportunities or share the proposal with corporate management to get funds for early development work

2. Early development

Rapid development phase to achieve the set goals, including:

- Adopt an appropriate OSS licence
- Release the first few versions of the project
- Create documentation to onboard new users
- Find early users within the broader community or company
- Author (or adopt) a Code of Conduct and enforcement strategy
- Improve the project quickly based on user feedback

3. Community Growth

As a community develops around the project, contributor teams:

- Set up community forums and community governance models
- Work on community engagement strategies to support the community
- Promote the project broadly at conferences, podcasts, newsletters, and more
- Expand and update the community documentation, especially for various kinds of contributors
- Continue to improve the project based on community feedback
- Continue to make releases in quick succession
- Focus on certain maintenance activities in addition to enhancement tasks

4. Maturity

Project reaches a steady state of growth with a dedicated community, characterized by:

- A linear or plateaued community growth
- Advocates for the project outside the immediate community
- A regular release cadence and release process
- Several active downstream projects
- Maintenance prioritized over feature development

5. Wind down

Eventually, project experiences community stagnation or decline

Project winds down activity and decides to archive, shut down, or transfer the project to a different space for continued maintenance

Project spaces are updates and the community words towards graceful transition

🙋 Learner Question

Which stage of the OSS Life Cycle is napari in?

napari is being continuously improved and maintained, and several (new and existing) users and contributors are actively participating in the project. Therefore, it's in the community growth stage. 😉

Being a Good OSS Citizen as a Corporate Contributor

Caution needed ⚠️

- Historically, corporate organizations & OSS have a shaky relationship
- Business relationships, are not always compatible with open source projects and communities
- Org needs and motivations to participate in OSS are not always aligned with or communicated to the project

General Good Practices

- **Join the community:** Each community has unique participation modes and channels
- **Observe:** Soak in information about the project and its culture, norms, and expectations
- **Start small:** Start with small contributions and work your way up to more complex contributions
- **Include the community early and often:** When you are ready to start contributing substantially, discuss it with the community first. As you continue to participate in the project, keep the community informed about your progress and any changes in your plans
- **Contribute upstream:** This critical contribution expands your impact beyond a single project and benefits the broader ecosystem

Familiarize with Community Standards 📖

- **(New) Contributor guidelines:** End-to-end workflows, issue/PR requirements, development setup, style guides, and more
- **Communication guidelines:** Project forums, good practices, Code of Conduct
- **Maintenance guidelines:** Reviewing guidelines, release process, CI/CD, project management, community management, upstream/downstream strategy
- **Governance:** Enhancement proposal process, project leaderships
- **Roadmap:** Current priorities and future direction

(New) Contributor guidelines

- End-to-end contribution workflow for the project and specific requirements
- Guidelines for creating context-rich issues & PRs and the appropriate process to move the discussions forward
- Instructions for setting up local or personal development environments, documentation systems, and test infrastructure, along with project-specific helper tools
- Style guides, i.e., project-specific preferences, for authoring code, writing documentation, and using design elements

Communication Guidelines

- Various community forums and their primary purpose, like support questions, developer discussions, etc.
- Guidelines for good asynchronous communication, for example, adding relevant context, asking specific questions, using jargon-free plain language, expected time to respond, etc.
- Code of Conduct to follow in the community spaces

Maintenance Guidelines

- Issue triaging and PR reviewing guidelines, for example, using project labels or bots
- Release procedure and cadence
- Continuous integration / Continuous Deployment (CI/CD) workflows
- Guidelines for upstream and downstream collaboration
- Instructions for supporting project management, community engagement, etc.

Governance

- Guidelines for creating and discussing Enhancement Proposals
- Recorded project leadership structure, particular interests or working groups, member roles and responsibilities, and more

Roadmap

A roadmap usually contains the project's current priorities, and future direction is decided with input from the community.

While having a roadmap helps determine priorities and the project's direction, many projects do not have a detailed roadmap but use other methods like EPs, issue tracker, and milestones.

Building Community Consensus 🤝

A vital component contributing is building consensus within the community on the design and implementation of these tasks at every stage of your participation.

☝🏼 **Funding or significant investment, such as in-kind donations, should follow the community, not the other way around.**

Organizational Open Source Strategy

Effective open source collaboration trickles down from leadership.

1. Alignment with the project's roadmap
2. Transparency around the corporate interest (build trust)
3. Corporate contributors contribute to general maintenance efforts regularly

Contributions

Small size and/or small to moderate scope ⛰️

Large size and/or significant impact 🌋

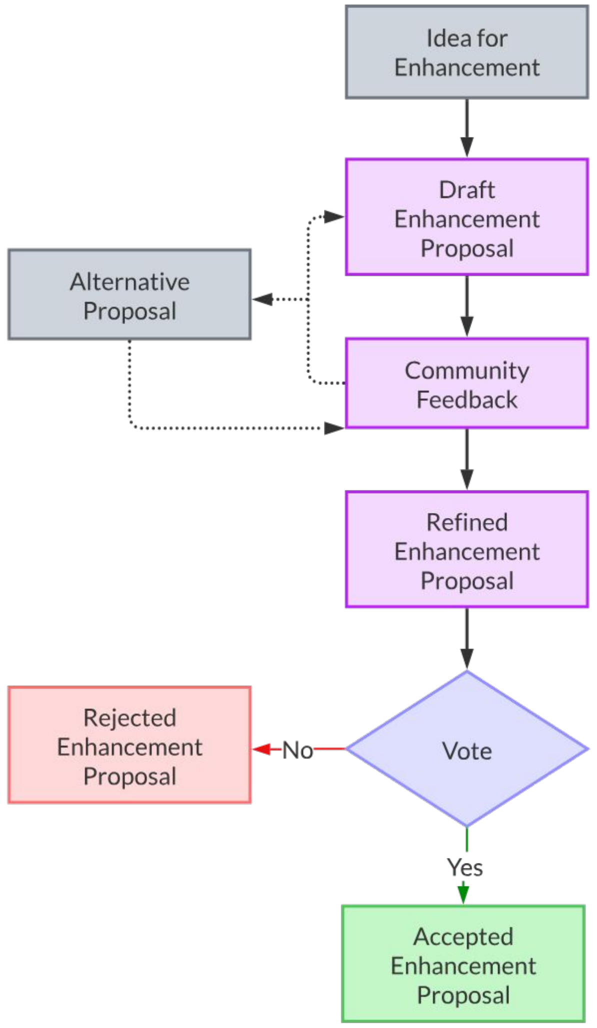
Considerations for Small-Scope Contributions ⛰️

Day-to-day decisions also need community consensus.

- Before complete implementations, discuss potential solutions on the issue tracker and open draft PRs for in-depth discussions when necessary
- Actively seek code reviews or alternative feedback mechanisms for non-code contributions
- Always consider the security, maintenance, and community implications of your work and discuss any concerns on contributor forums

Steps for Large-Scoped Tasks 🌋

1. Following proper EP guidelines
2. Work with 1-2 community members on the draft before broader community feedback
3. Keep an open mind for community feedback
4. Make appropriate updates to your proposal based on the community feedback
5. Sometimes the best strategy could be to withdraw the proposal and start afresh
6. Do not take a rejection personally


Idea for Enhancement
Draft Enhancement Proposal
Alternative Proposal
Community Feedback
Refined Enhancement Proposal
Vote
No
Rejected Enhancement Proposal
Yes
Accepted Enhancement Proposal

Turn Private Conversations into Public Resources 💬

One of the earliest points of friction is the difference in how internal vs. community consent-based communications happen, and it can be tricky to identify the right balance between the two.

🔗 [Full process, checklists, and templates in the GitHub repository]

Pre-requisites and Assumptions

- ❏ Your **private conversations are transcribed** (or can be) or documented (perhaps through internal agenda documents).
- ❏ You are **starting from a place of trust**.
- ❏ You know and **understand your community's guidelines** (or why none exist).

The Process 🪜

1. **Setup:** Create a shareable record of the conversation, keep a list of everyone involved, communicate the desire to for public discussions
2. **Consent & Credit:** Draft a message seeking consent and review, share the message along with the document, record responses
3. **Loop back with participants:** Update document based on consent response, share updated resource with participants, incorporate feedback
4. **Share with the community:** Give participants a heads-up, and share with the broader community

🔗 Full process, checklists, and templates in the GitHub repository

Tips for transparent communication 💡

- Project management: favour open tools and practices
- Synchronous meetings: follow consensus guidelines
- Asynchronous discussions: follow the community’s lead

⚠️ **Ensure that all stakeholders are comfortable with the open workflow before implementing it.**

Examples of Good Corporate Open Source Contributions 🖼️

- CPython Developer-in-Residence Program (PSF Sponsors: Google, Bloomberg, and Meta)
- Faster CPython Initiative (Microsoft)
- Open Sourcing Memray (Bloomberg)
- Consulting with Open Source Projects (Quansight)

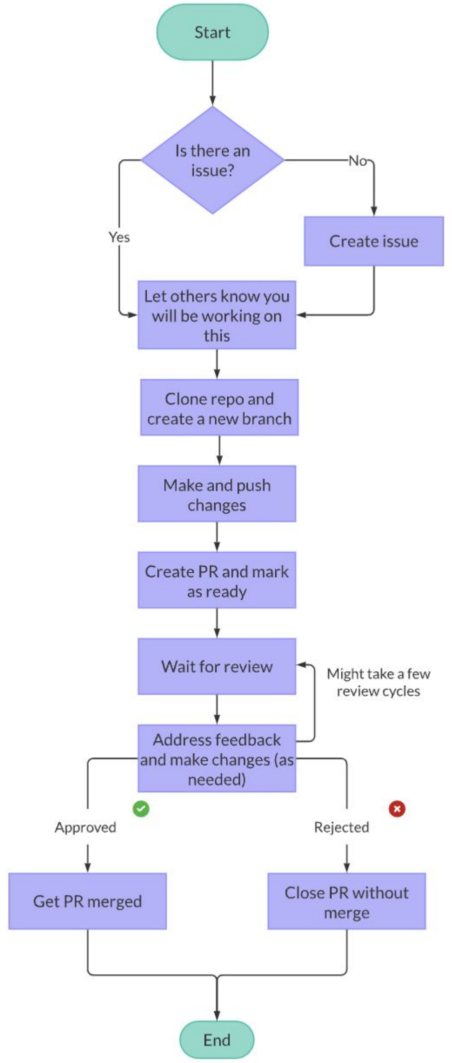
Break ☕

Contributing to Open Source

How contributions work 🎁

In general, OSS projects allow individuals to make contributions in the form of **issues or Pull Requests**.

Community members can also perform **reviews and provide feedback or suggestions through comments** on the ideas and changes which they or others have suggested.


Start
Is there an issue?
No
Yes
Create issue
Let others know you will be working on this
Clone repo and create a new branch
Make and push changes
Create PR and mark as ready
Wait for review
Might take a few review cycles
Address feedback and make changes (as needed)
Approved
Rejected
Get PR merged
Close PR without merge
End

Contributions beyond code ✨

OSS projects also need contributions in **documentation, design, marketing, community management, fundraising, legal counsel, accessibility auditing and remediation work, translation and localization, and more.**

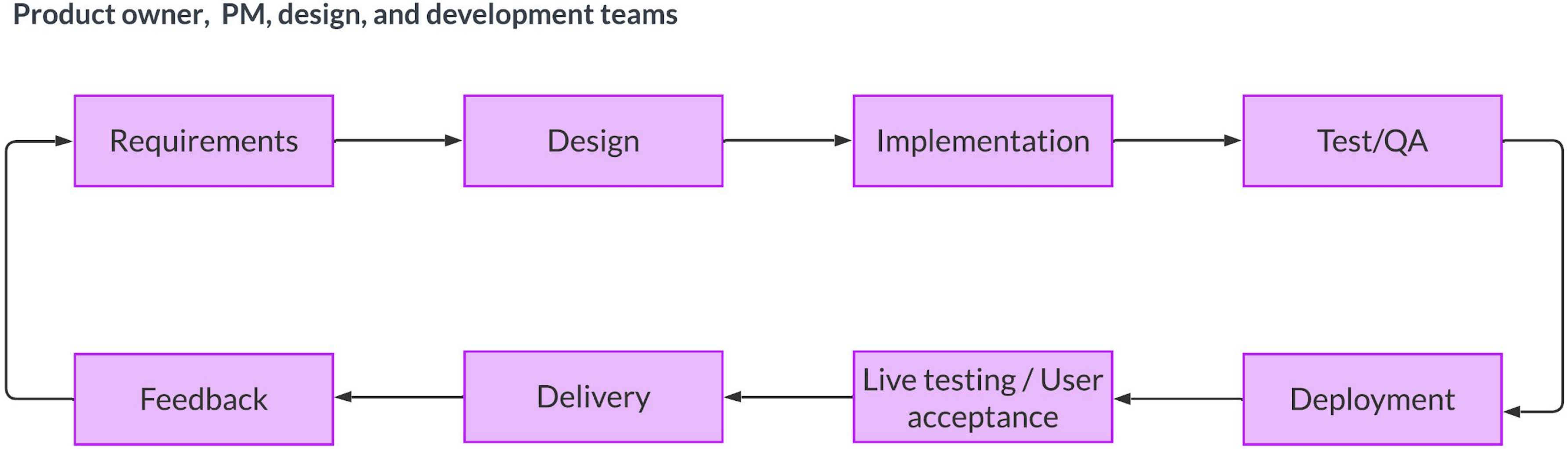
These contributions are as significant as code contributions and are often called **code-adjacent contributions.**

Getting the most of the issue trackers 🎟️

- **Scan the issue tracker for existing issues & pull requests** before opening a new one
- If there is not an issue, aim to **provide enough information** for the triagers or maintainers to assess your issue
- Give your issue a **meaningful title**
- If the project has **issue templates, use them**
- Be **kind and thoughtful**
- You can also contribute by giving feedback or **adding more information to existing issues**: document missing elements, cross-link relevant discussions, summarize long discussions

Development and maintenance considerations 💻

Agile Life Cycle: Product-focused software development methodology


Product owner, PM, design, and development teams
Requirements
Design
Implementation
Test/QA
Deployment
Live testing / User acceptance
Delivery
Feedback

Assumptions in Agile

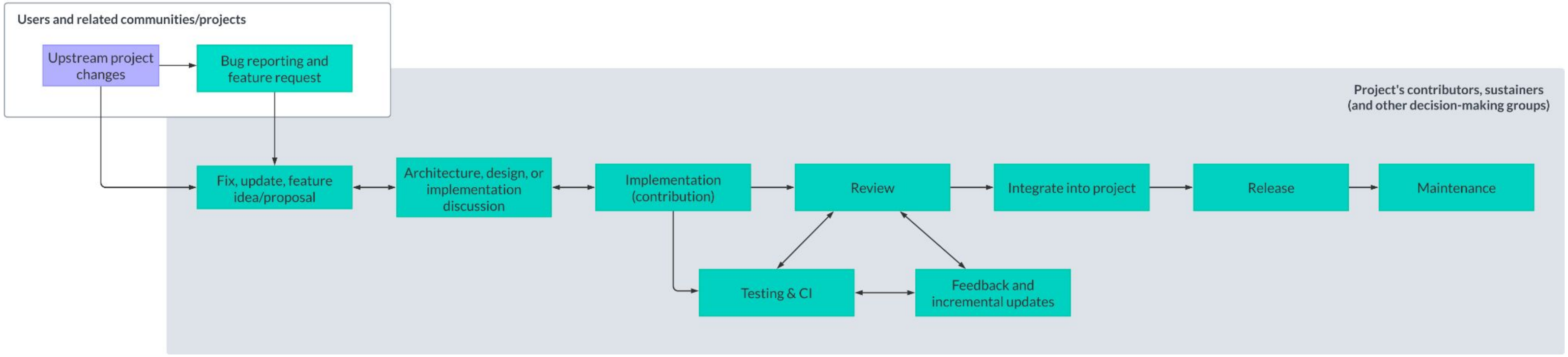
- **Dedicated team** of developers, designers, and project/product management
- Clear **distinction between the development team and the product's end-users**
- There is a **project manager/owner**, responsible for determining the scope of work, prioritizing tasks, and ensuring the team meets its deadlines
- Product goals are often **tied to business goals** and are measured by revenue, user growth, ROI, and more metrics

Assumptions in Agile (Contd.)

- Depending on the licensing and revenue generation model, the organization might also be responsible for providing **support to end-users** and have an SLA (Service License Agreement)
- Even if using open source tools or software to build the ultimate product, the product itself may or might not be open source. Thus, the **ultimate decision-maker is the product owner**; no consent or consensus is sought
- Due to the time and delivery expectations, this **isn't easy to implement with a fully geographically-distributed team** that communicates asynchronously

Development and maintenance considerations 💻

The OSS Way: Collaboration-focused development methodology


Users and related communities/projects
Upstream project changes
Bug reporting and feature request
Project's contributors, sustainers (and other decision-making groups)
Fix, update, feature idea/proposal
Architecture, design, or implementation discussion
Implementation (contribution)
Review
Integrate into project
Release
Maintenance
Testing & CI
Feedback and incremental updates

OSS vs. product development methodologies 🔎

Open source development

- Remote teams and asynchronous communication
- Consensus-seeking, co-ownership (peer review, collaboration) and transparency focus
- Goals are dictated by the community
- Deadlines are rarely set in stone
- Scale up and down depending on community availability
- Resilient to organizational change

Product development (such as Agile)

- Dedicated team and synchronous communication
- Product owner makes final decisions
- Goals tied to business goals measured through ROI, revenue, etc.)
- Project owner and manager align priorities, timelines and keep team aligned
- Clear distinction between development and end-user

Characteristics of OSS Development

- Designed for **geographically distributed** teams and **asynchronous communication**
- OSS maintenance and development is **based on transparency, trust, collaboration, and co-ownership**
- Relies on **distributed decision-making and consensus-seeking**
- Contributions are **peer-reviewed**
- **End-users and community members** equally provide feedback, raise bugs, and request features
- The project's goals are **tied to the community's goals** and values
- Operate through processes designed to be resilient to organizational change and allow for **self-organization and self-management**

Characteristics of OSS Development (Contd.)

- While projects usually have a release cadence, **deadlines are rarely set in stone**, and a significant release might be delayed if there are blockers or the community needs more time to review and test the release
- The development **model can scale (up and down)** depending on community participation and maintainers' availability
- **Trust is built by past records of productive participation** and wise decisions on smaller issues
- A common mantra in OSS is **"Release Early, Release Often"** (refer to the schematic above)

Designing in the open 🤔

- **Communicate early and often** on the project's preferred communication platform
- Anticipate feedback and **proactively provide context** and rationale for your design decisions
- **Acknowledge given feedback** and re-work your contribution
- Signal **willingness to adapt your design** if someone else is willing to collaborate with you
- **Plan for modularity**, even if the first designs are not modular

Working with large distributed teams 🌎

Community: A group of people united by a common identity and collective purpose and who collaborate by sharing ideas, knowledge, and resources.

In OSS:

- No two communities are the same
- Communities do not belong to or work for a single entity

So, how do you engage and work with an OSS community? 🤝

What can you offer? Depending on your (individual, team, or organization) capacity, you might be able to contribute in different ways, for example, through sponsorship, in-kind contributions, or making direct contributions through your teams or individuals.

Determine your time commitment. How much time can you dedicate to the project? Is it a one-off contribution or a long-term commitment?

How much do you know about the community? Spend time researching how the community communicates (e.g., identify their platforms and lurk there a bit to get a sense of their communication standards). Also, try to understand the community's governance model and decision-making processes.

Community Engagement Tips 🤝

1. **Introduce yourself.**
2. **Communicate what you are working on.**
3. **Give back.**
4. **Plan an exit strategy.**

🔗 *Helpful tools for contributing to OSS available in the material*

03. Impactful OSS contributions

Creating a New OSS Project

New OSS Project Motivations and Checks ⚡

- Is the project idea beneficial to a substantial community of users?
- Is there an existing community OSS project implementing a similar solution?
- What is your core motivation to develop this project under an open source license?
- Can you dedicate resources to build and maintain the project sustainably?
- Are you prepared to collaborate with the open source community?

Preliminary Considerations 🧱

- Open Source Values
- User Workflows
- Project Roadmap
- Preliminary Governance
- Licenses and CLAs
- Stakeholder Review
- Project management

OSS Development and Maintenance 🏗️

- OSS Development Practices
- Documentation
- Maintenance
- Upstream Contributions

☝🏼OSS development guidelines apply through most of your OSS life cycle and will take different forms as your project matures.

Community Engagement 🌱

- Code of Conduct (CoC)
- Governance
- Messaging and Brand
- Communication Channels
- Advocacy
- Partnerships

Exit Strategy 🏛️

- Exit Conditions
- Communication
- Knowledge sharing
- Onboard and *offboard* leaders

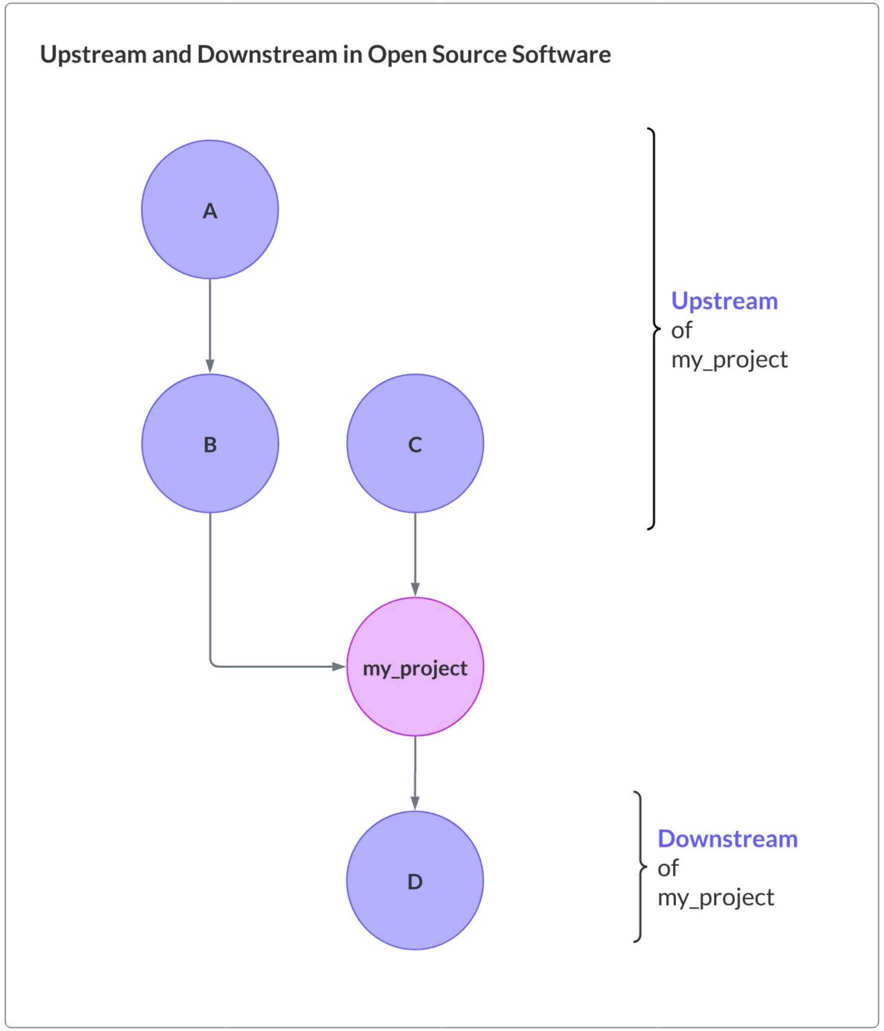
✨ **This is not a plan for failure, but a path to building a project and community that can thrive independently and eventually make space for newer projects.**

Effective upstream & downstream collaboration

Understand Upstream & Downstream Projects ↕️

Refers to the relationship between different projects that build upon or contribute to each other

Related software development terms are “dependency” and “dependents”



Why Contribute to Upstream Projects? 💭

- Good OSS Citizenship
- Evaluate ideas with community experts
- Avoid duplicated effort
- Project stability

Upstream Engagement Strategy 🌳

Cardinal rule: Upstream first

- **Critical projects**: active contribution, maintenance, community participation
- **Supporting projects**: feedback, contributing back as needed

Creating an impactful Open Source strategy

🙋🏽‍♀️ Learner checkpoint

Does everyone in your organization or team:

- know what licenses they can use to create or consume products?
- know if you have policies on releasing code as open source?
- know who to contact if they have questions about open source?

Where to start? Defining your open source goals 🎯

1. Understand your organization’s mission & goals
2. How does OSS support that mission & goals
3. Decide and document OSS strategy
4. Design an implementation plan

☝🏼 If your organization's goal is to **"enable the universal and immediate open sharing of all scientific knowledge, processes, and outputs"** then there is a clear link to the principles of Open Research and Open Source.

Components of an open source strategy

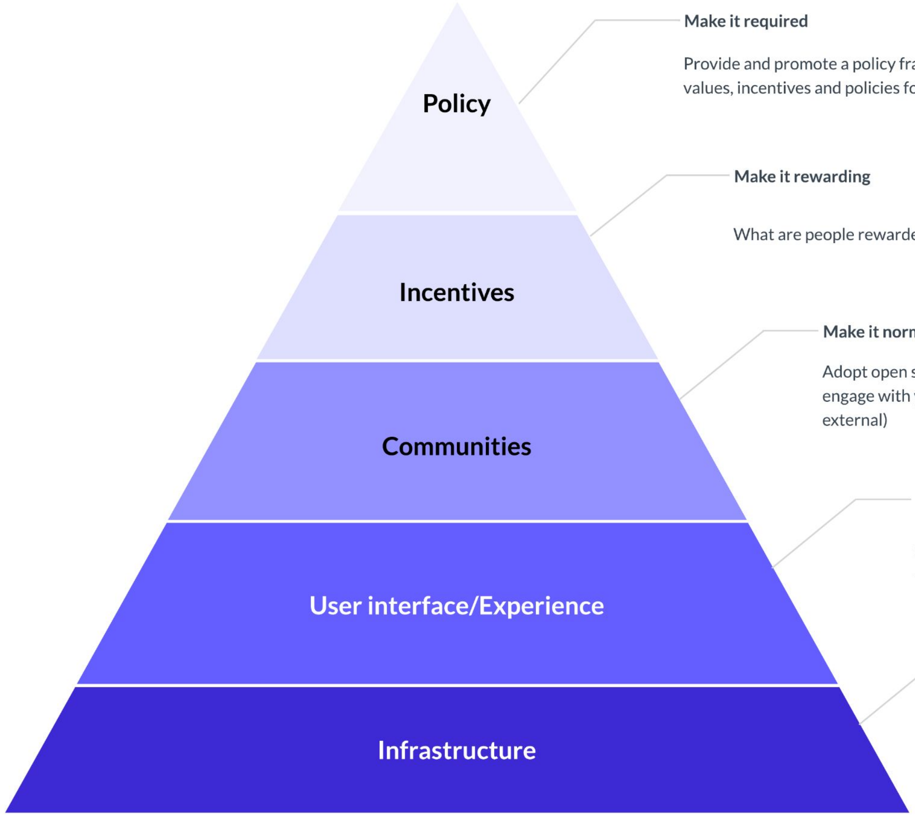
1. Core principles: your organizational approach to OSS
2. Key stakeholders: how do we know which all to focus on and for whom?
3. Governance: get everyone on the same page
4. Organization-wide processes and policies:
   a. OSS development and consumption
   b. OSS maintenance, licensing, compliance, security
   c. Community engagement

Guiding principles for open source strategies and contributions 🧭

1. Engagement starts early
2. Community comes first
3. Focus on transparency
4. Align with community standards
5. End gracefully

✨ See suggested tactics for each principle in the repository

Implementing OSS strategies and programs 🌱



Make it required

Provide and promote a policy framework for aligning values, incentives and policies for the organization.

Make it rewarding

What are people rewarded for doing? Align incentives

Make it normative

Adopt open source participation practices and engage with your communities (internal and external)

Make it easy

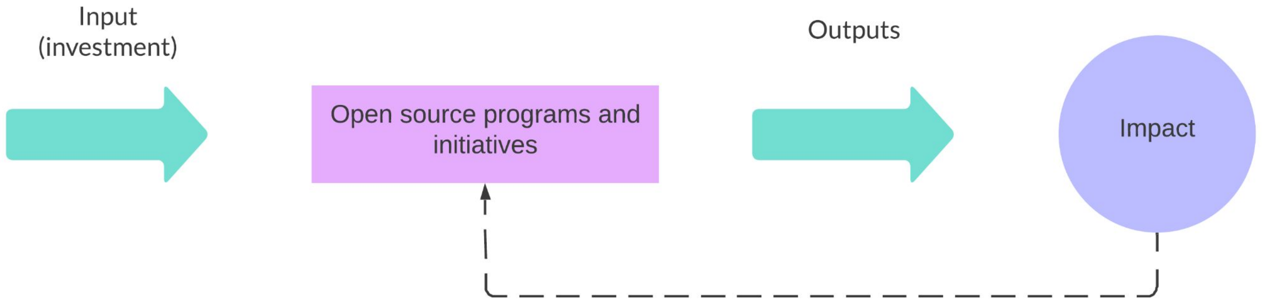
Streamline processes and increase transparency

Make it possible

Enable individuals and teams within the organization to participate in open source communities

Measuring Success of OSS Strategy 📈

- Identify + track your investments
- Identify + track impact
  - How do your stakeholders define success?



☝️ A general suggestion when measuring investment or impact in open source is avoid estimating ROI (return on investment) for your open source efforts.

Choosing better metrics

Identify objectives

Define observable aspects

Establish a baseline

Define your desired trend

Determine signal phenomena

Activity:
OSS Community Map

Team Activity: OSS Community Map (15 mins)

- You will be split into breakout rooms of 3-4
- Grab a copy of the activity worksheet (link in the chat)
- Work on your community map (Tania will share an example in a minute)
- Finally, we will reconvene as a group to share any discoveries or ask questions

✨ Remember to refer to your Day 1 power map

Resources
What resources do the key interested parties value?

- Time
- Skills and expertise
- Treasure (financial contributions, grants and other funding)
- Human infrastructure
- Open and widely accessible tools and infrastructure
- Documentation, tutorials, how-to's

Disruptions
Who/what can disrupt the current system? (e.g. loss of resources or key parties)

- Over-representation/single representation within decision-making groups
- Non-OSS/community friendly corporate backed alternatives
- Inaccessibility of the tools*

Key interested parties
Who is involved in the project? Who maintains or advances the project and status quo?

Which of these are system critical? 🛠️

- End users (researchers, educators, developers, data scientists, journalists)
- Organizations as users (universities and research institutions, government agencies, non-profits)
- Corporate sponsors/contributors (Quansight, QuantStack, AWS, Bloomberg, IBM, 2i2c)
- Many many projects*
- Project contributors, core teams 🛠️
- Software steering council, executive council 🛠️
- Distinguished contributors 🛠️

https://jupyter.org/governance/intro.html

Power and influence
Who has control over the resources? Who (or which groups) holds the greatest influence? 🛠️

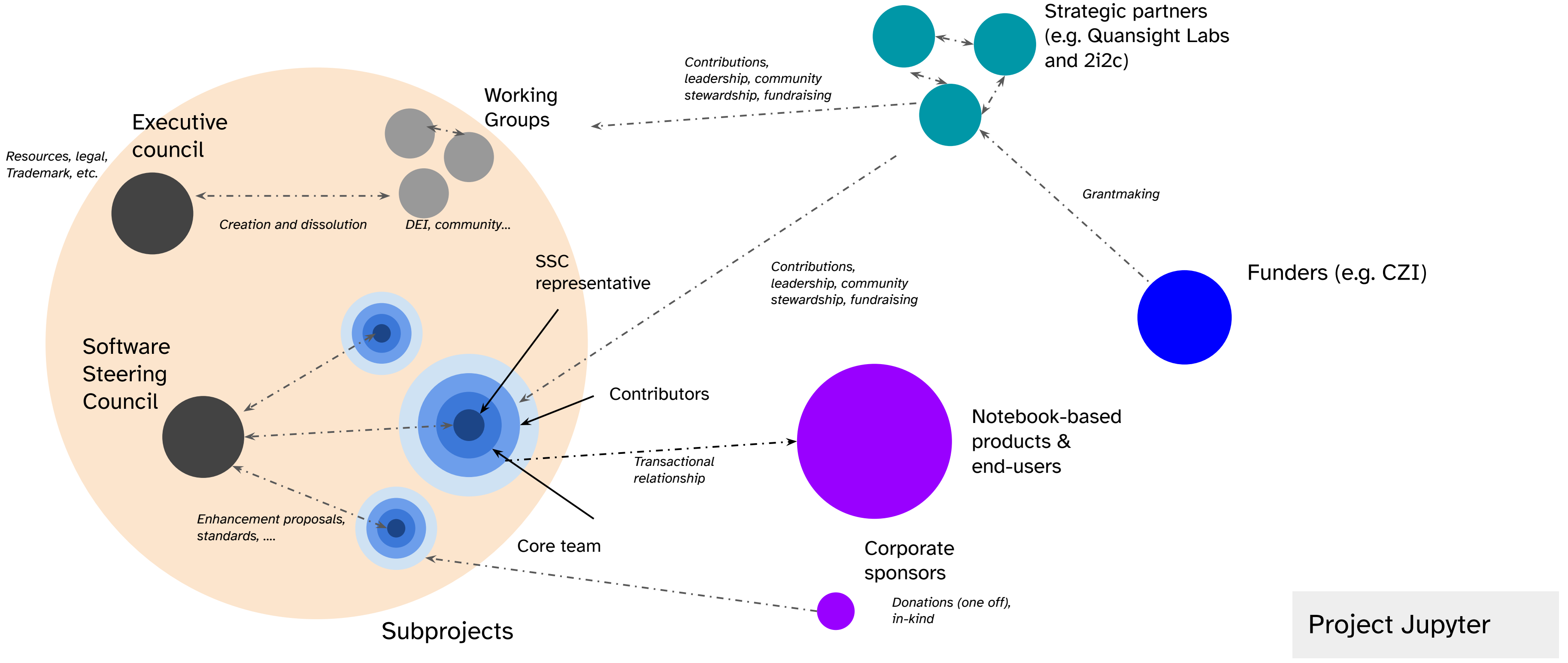
- Project contributors, core teams 🛠️
- Software steering council, executive council 🛠️
- Distinguished contributors
- Working groups

https://jupyter.org/governance/intro.html

Jupyter

Community mapping

Building from the power map activity build a project community map. In this activity we aim to establish the relationship between stakeholders parties, the resources and power dynamics.
Make sure to include the different types of interested parties (including your organization), how they interact with each other and the project itself (i.e. what they contribute with and the relationship direction).


Strategic partners (e.g. Quansight Labs and 2i2c)
Contributions, leadership, community stewardship, fundraising
Working Groups
Executive council
Resources, legal, Trademark, etc.
Creation and dissolution
DEI, community...
Grantmaking
SSC representative
Contributions, leadership, community stewardship, fundraising
Funders (e.g. CZI)
Software Steering Council
Contributors
Notebook-based products & end-users
Transactional relationship
Enhancement proposals, standards, ...
Core team
Corporate sponsors
Donations (one off), in-kind
Subprojects
Project Jupyter

Thanks for your attention!

Thanks for your attention!

🗳️ **Feedback survey: bit.ly/oss-survey-w**