Evaluating Open Source Software Projects

99-520 Summer 25

First Presentations

Team presentations on 18 June

10 Minutes per team

Put together a few slides

Everyone on each team presents

Tell us about …

- The open source project your team is working on. (How old is it? What do people do with it? How big is it? etc.)
- The team project you are undertaking together. What is your understanding today about the work in front of you?
- Team status with respect to mentor meetings, tool chains, boot-strapping into the work, how your team is organizing the work, etc.

Keeping Notes

As you work in your part of the project, start keeping an electronic log

Choose any note taking app (Google Doc, MacOS notes, notepad, etc.)

Cut-and-paste as you go – commands, output, observations, profanity

This will help you:

- maintain and share context,
- debug problems with others (e.g., mentors, each other)
- teach/mentor others (and note differences too!)

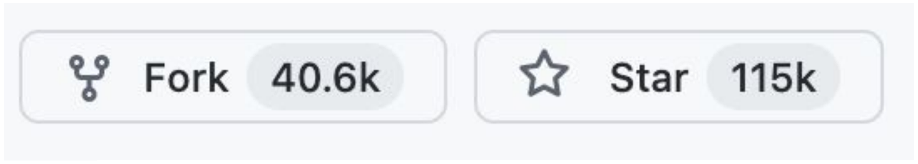
SBOM Example Notes

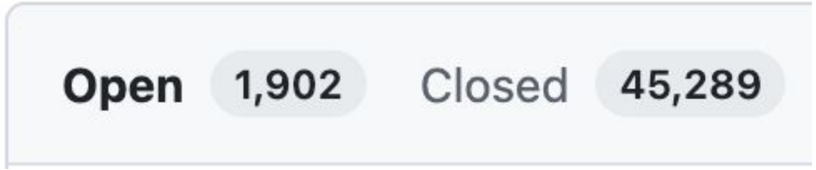
How do you measure open source project success?

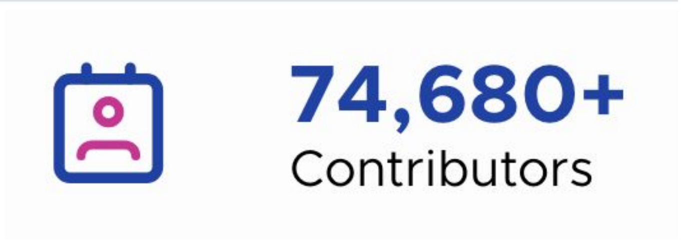
What does success mean?

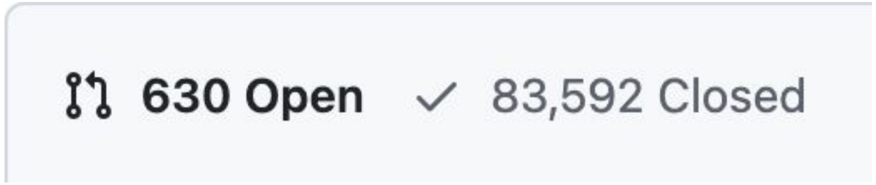
- GitHub Stars!
- Downloads! Forks!
- Bug Reports & Issues
- Number of Contributors!

- **Code contributions (commits)!**









What does success mean?

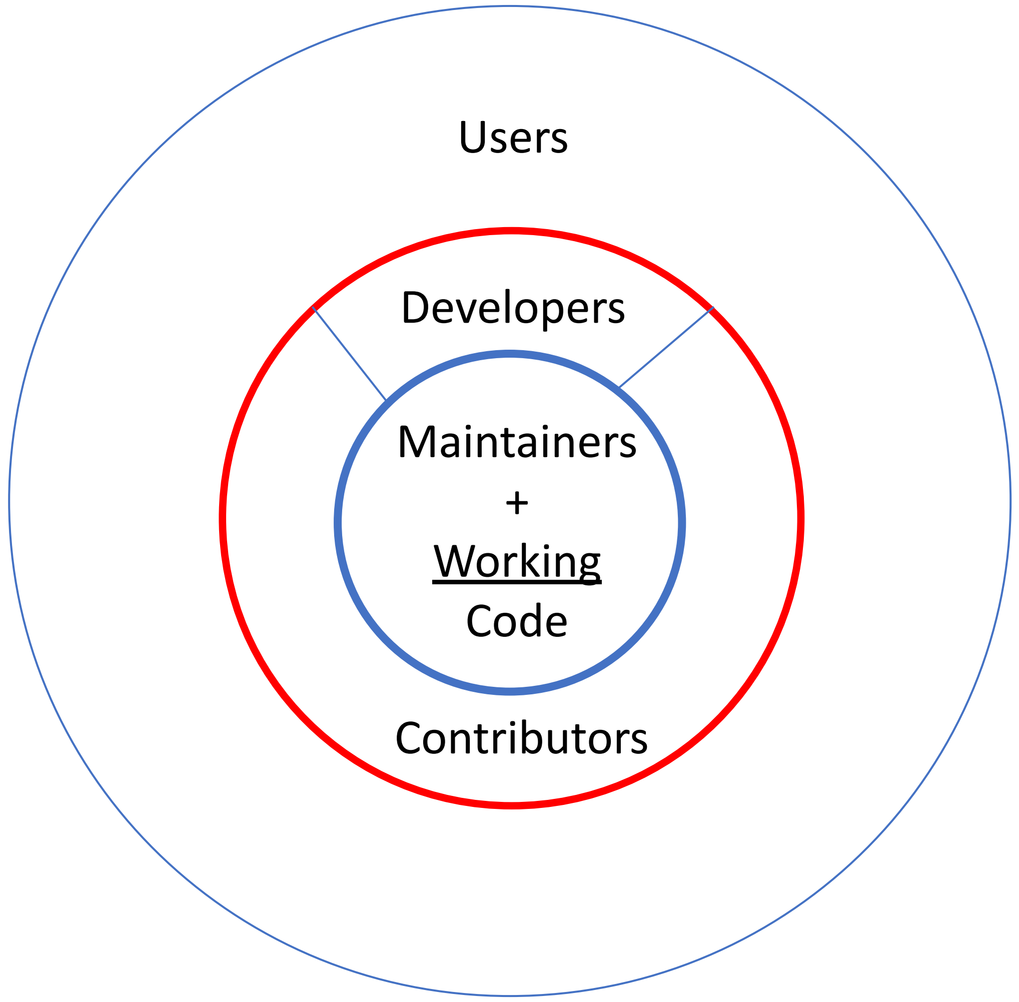
- Outbound
  - Did someone star a GitHub repo because they are a simple end user, or because they follow the developer, or because they are a developer using the source?
  - Did someone clone/fork a repo and use it, or abandon the project fork, or simply review and learn from it?
  - Download counts has similar problems.
- Inbound
  - **Measure actions, activities, and artifacts that mean something was done**
  - A healthy contribution flow of all work

Let’s focus on project structure to build those flows

Roles

- **Maintainer**: A primary author of a project with full privileges to write the project directory tree. Typically, the creator. (AKA **Committers**)
- **User**: Any person that is using a software project for its intended purpose. (Users are essentially invisible until they contribute. A fork or download tells the project nothing!)
- **Developer**: Any person using a software project but additionally modifying the project **to their own needs**. (Still invisible until they contribute.)
- **Contributor**: any person offering a direct artifact back to the project, including source code patches, bug reports, configuration info, documentation of any kind, including presentations, articles, blog posts, etc.

Roles



What Sort of Contributions Can People Make?

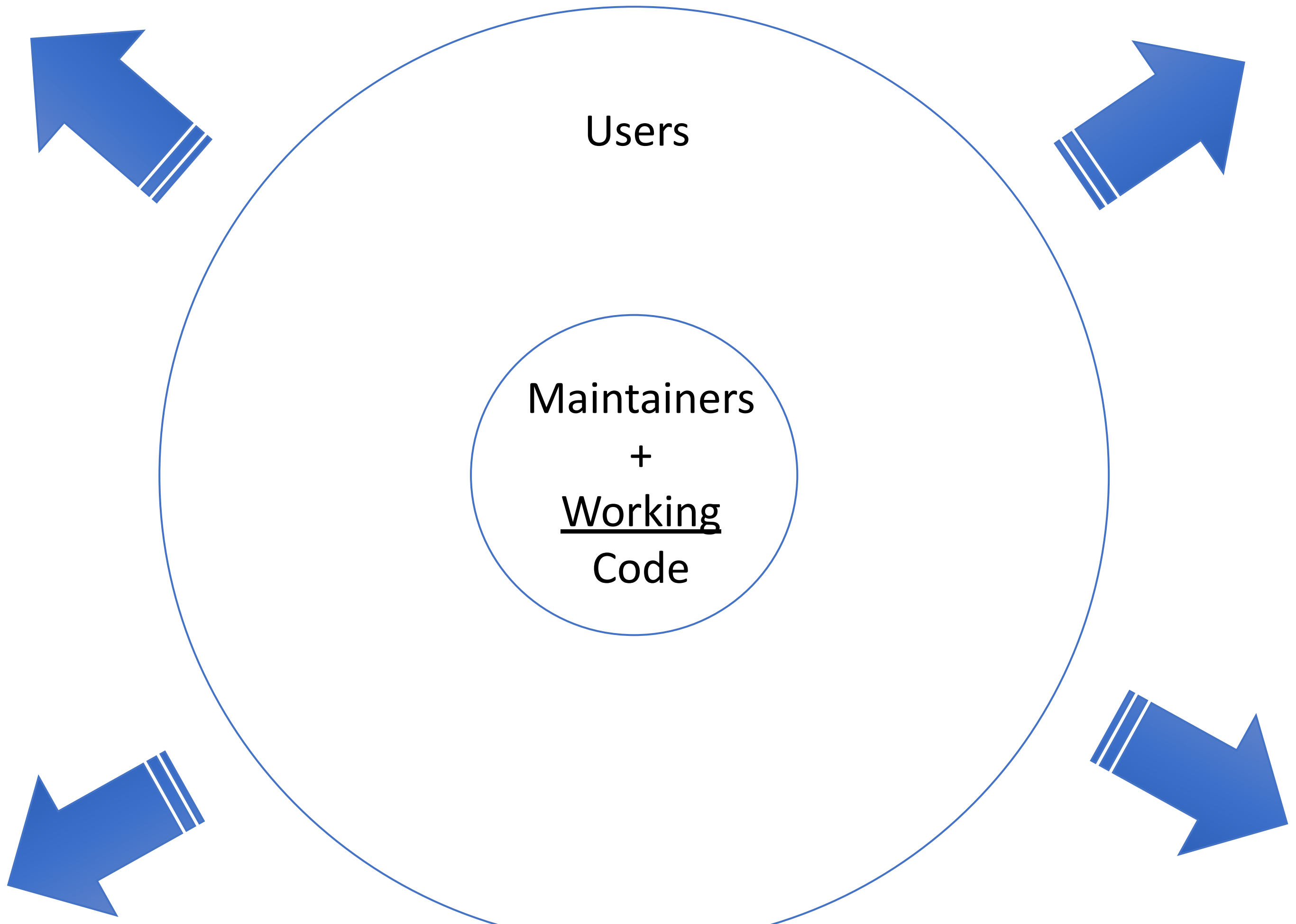
Contributions

- Code – fixes and functionality increases code value and broadens use
- Bug reports indicate a new test path or use
- New configurations broadens the user base increasing use value
- Documentation (answering forum/email questions, creating tutorials, presentations, FAQ, etc.) broadens the user base
- Forum <u>time</u> answering questions
- Translations (messages, user document, etc.) can make a big difference
- ANYTHING SOMEONE BRINGS TO THE PROJECT

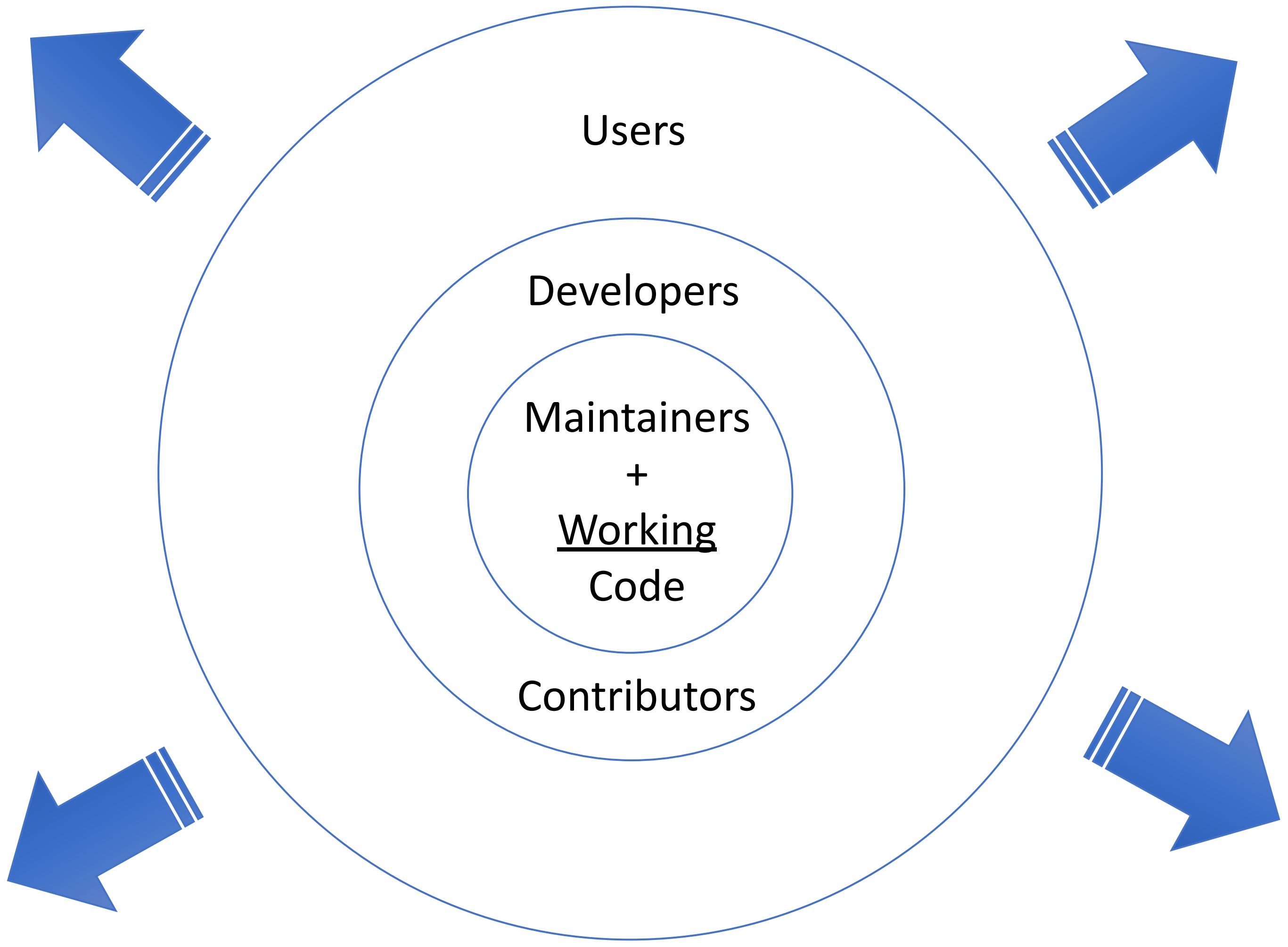
Three On Ramps Need to be Built

Maintainers
+
Working
Code

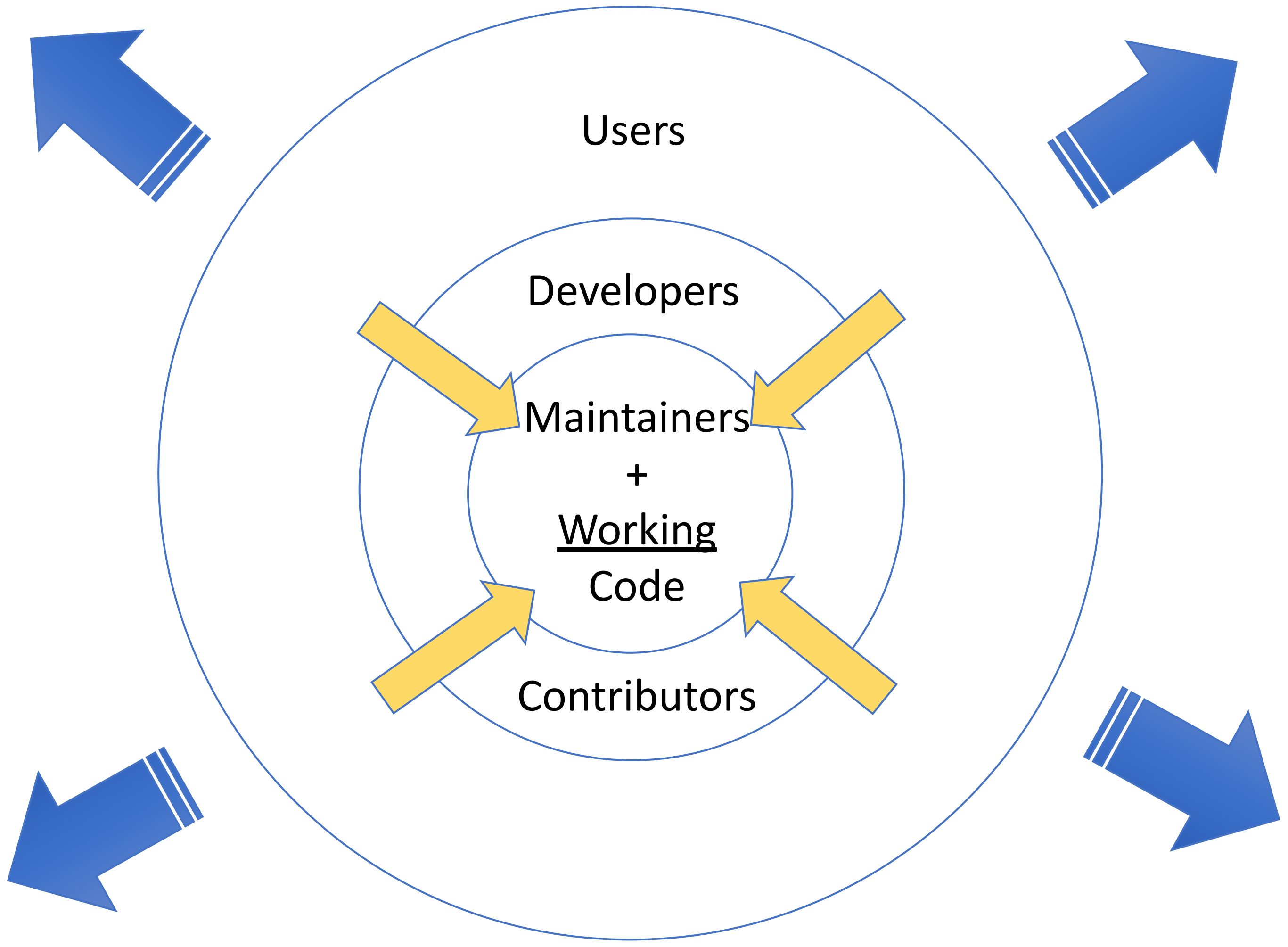
Three On Ramps Need to be Built



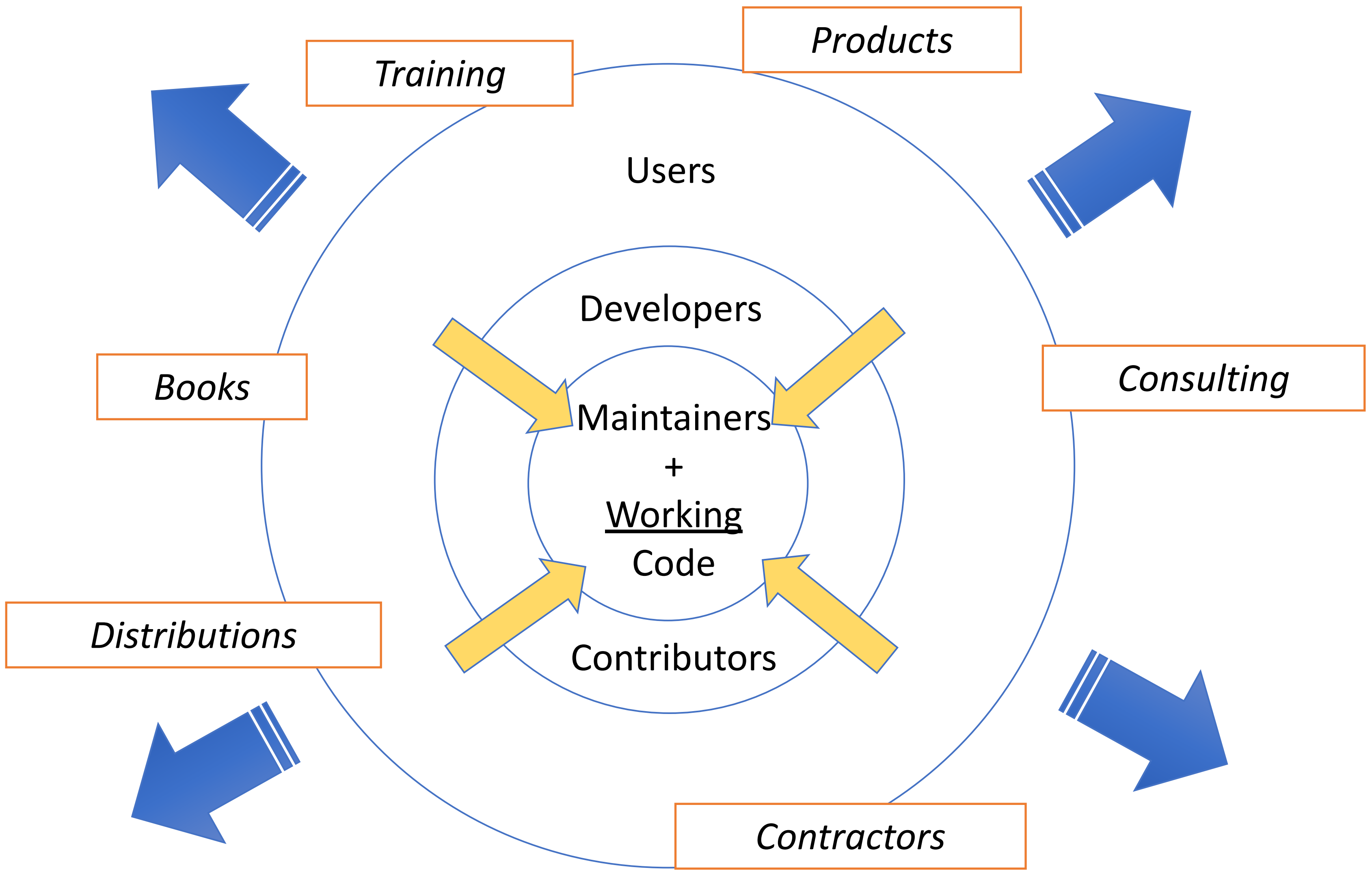
Three On Ramps Need to be Built



Three On Ramps Need to be Built



Three On Ramps Need to be Built



Three On Ramps for Community Building

How do you encourage people to use our project?

(Because that is where we will find bugs reports & developers)

How do you encourage developers selfishly to experiment?

(Because these are our future contributors)

How do you encourage developers to share their work?

(Because contribution flow is the growth and success of our project)

Three On Ramps for Community Building

How do you encourage people to use our project?

(How do we make it easy to install/configure/use the software?)

How do you encourage developers selfishly to experiment?

(How do we make it easy to build/test/experiment?)

How do you encourage developers to share their work?

(How do we make it easy to contribute?)

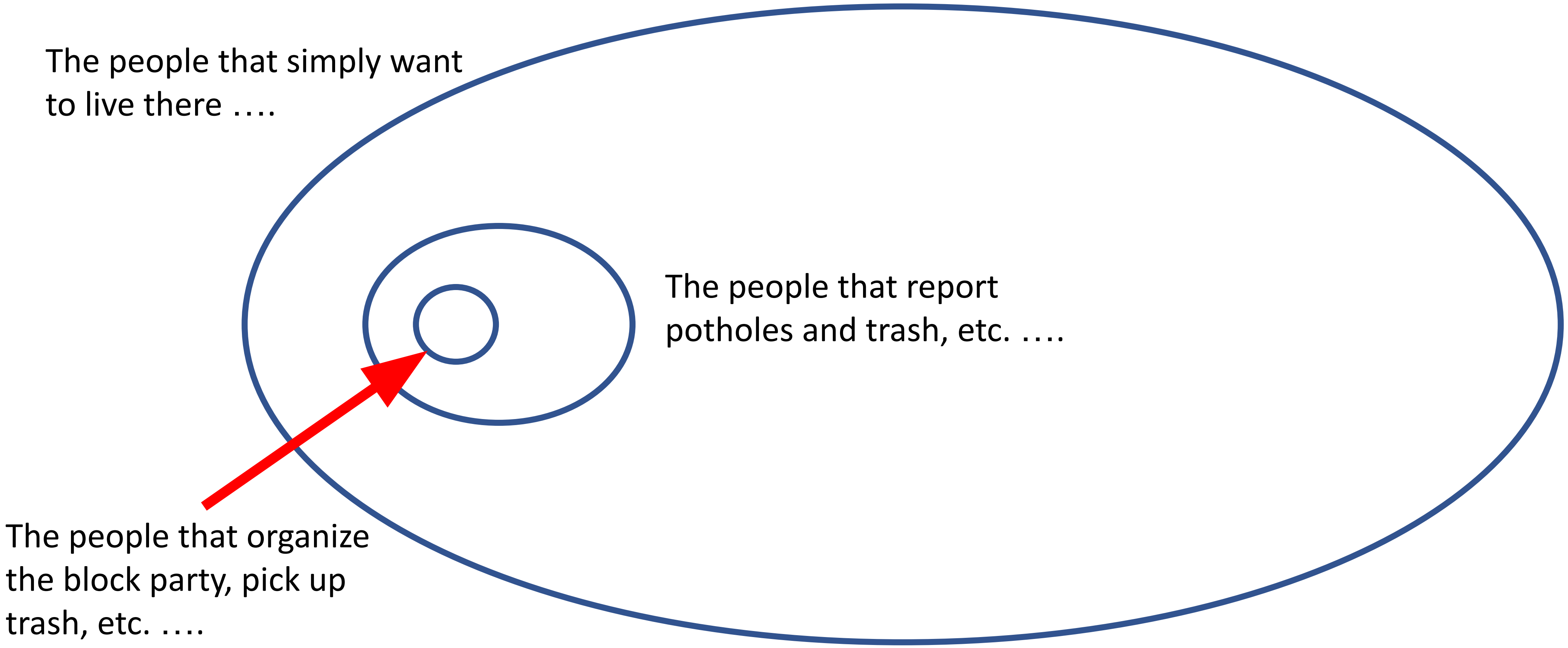
[Cynical] Side Notes:
“There is no community except for the one you build.”

- Nobody cares about your community
- Your project solves a problem for a user or a developer
- They want to selfishly use your work … that you liberally licensed to allow them to do just that.
- You can have <u>NO</u> expectations in return
- It is your job as maintainer to make it easier to do the right thing economically and contribute back to the project

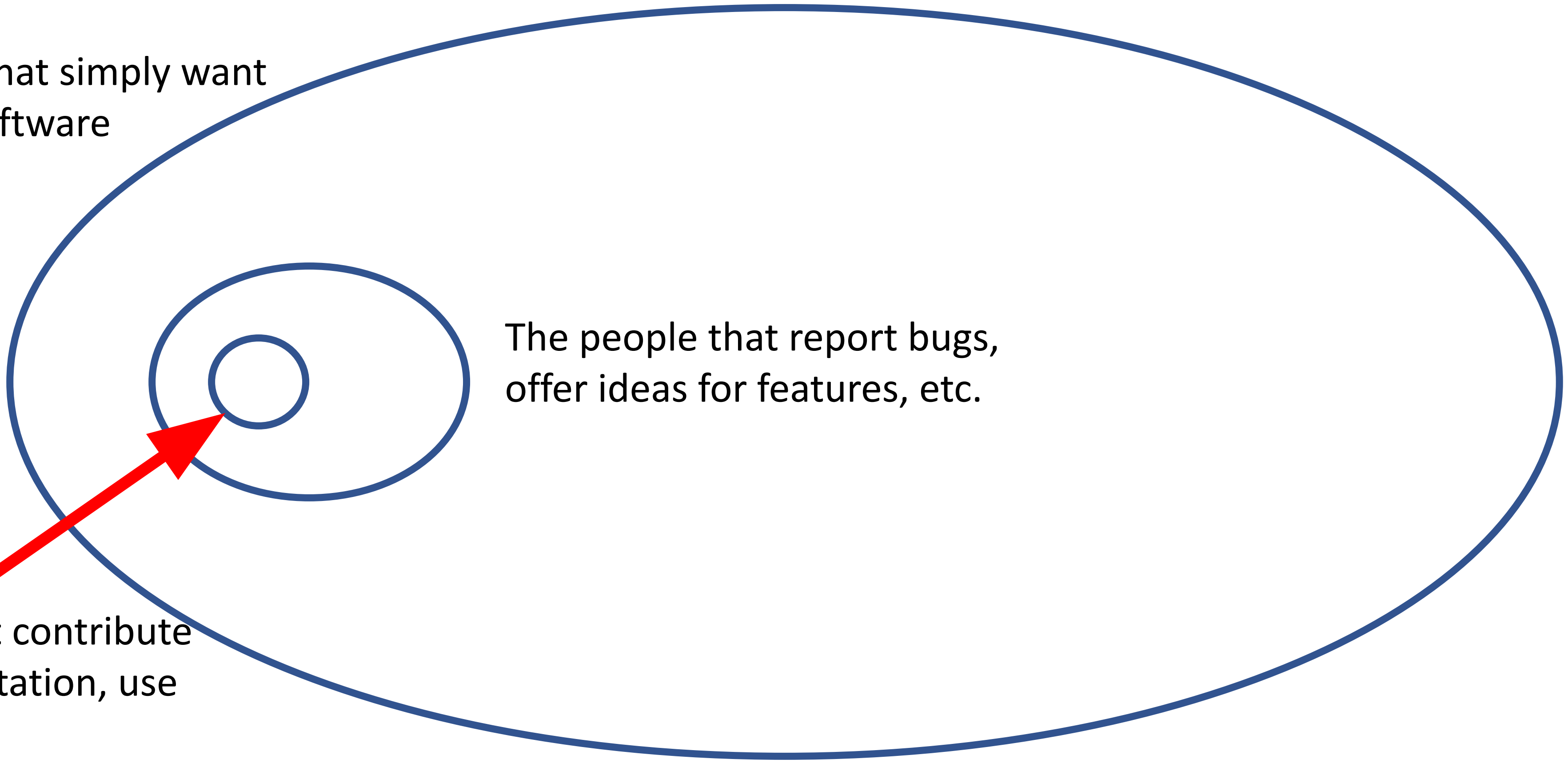
What do we think about Freeloaders?

In the World of Atoms: You choose your neighborhood for very personal reasons

Three Sorts of Neighbours in Your Community


The people that simply want to live there ….
The people that report potholes and trash, etc. ….
The people that organize the block party, pick up trash, etc. ….

Three Sorts of People in Your Project Community

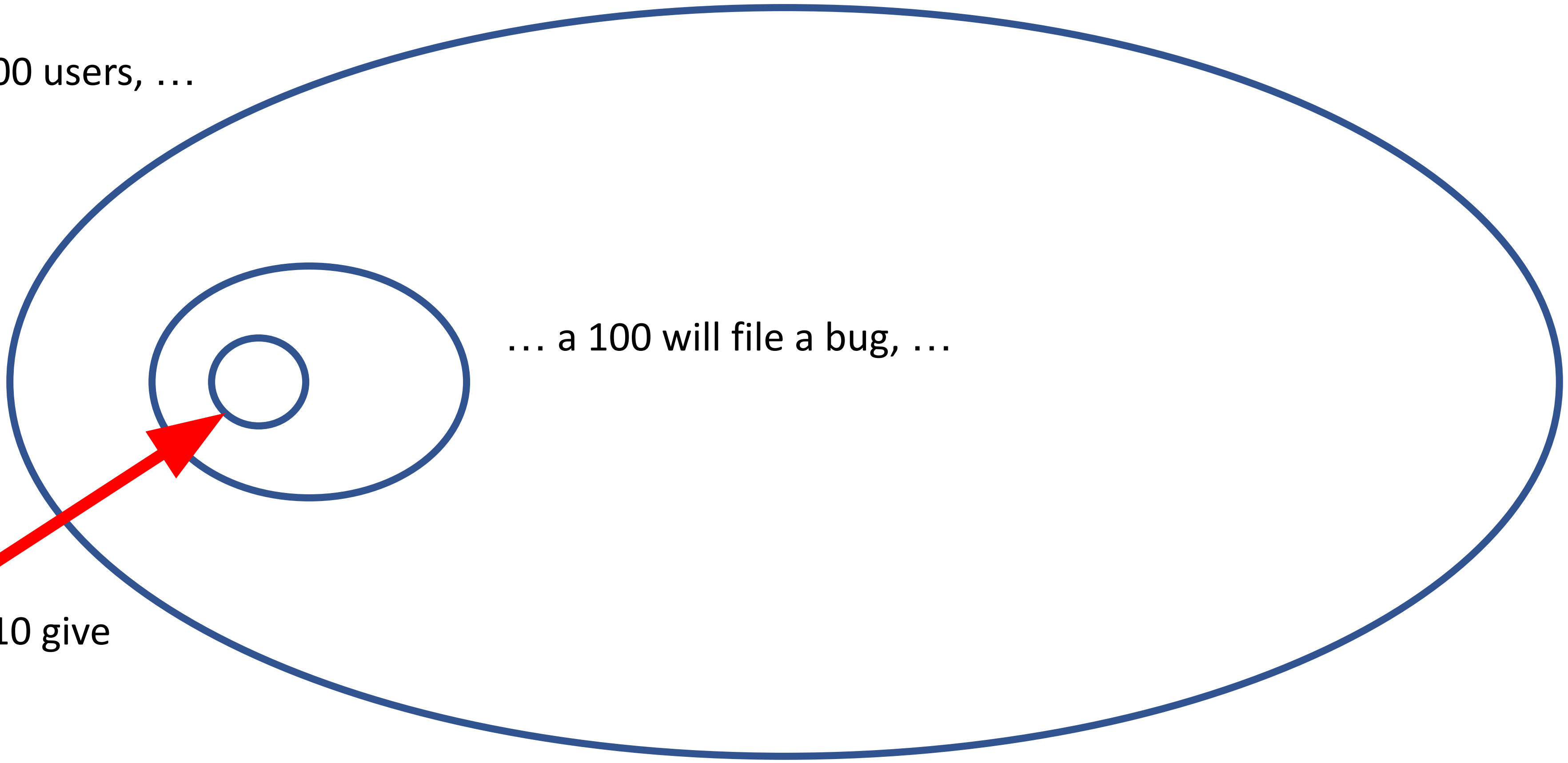


The people that simply want to use the software

The people that report bugs, offer ideas for features, etc.

The people that contribute code, documentation, use cases, etc.

Rules of Thumb and Orders of Magnitude



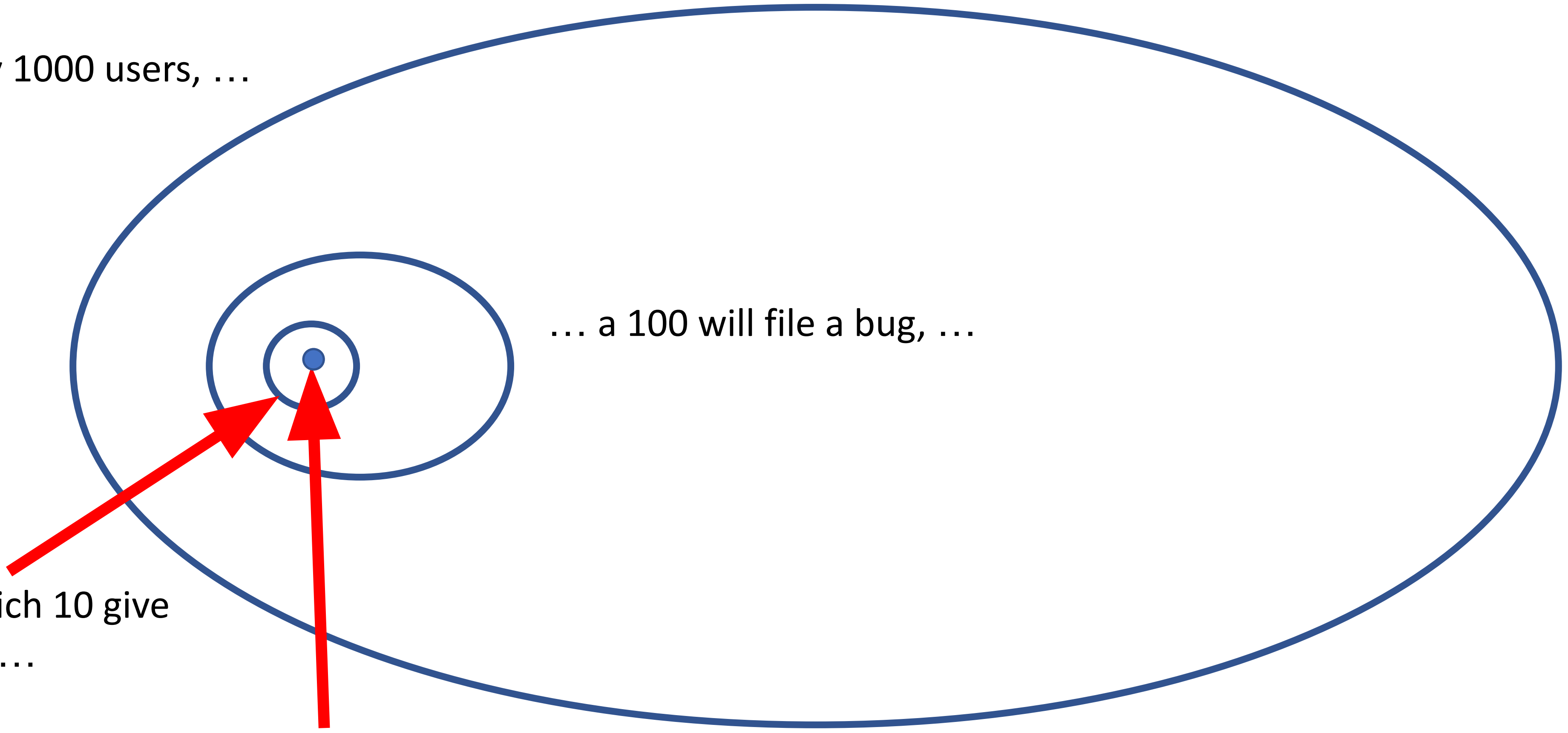
For every 1000 users, …

… a 100 will file a bug, …

… out of which 10 give you a patch, …

Rules of Thumb and Orders of Magnitude (Again)

For every 1000 users, …



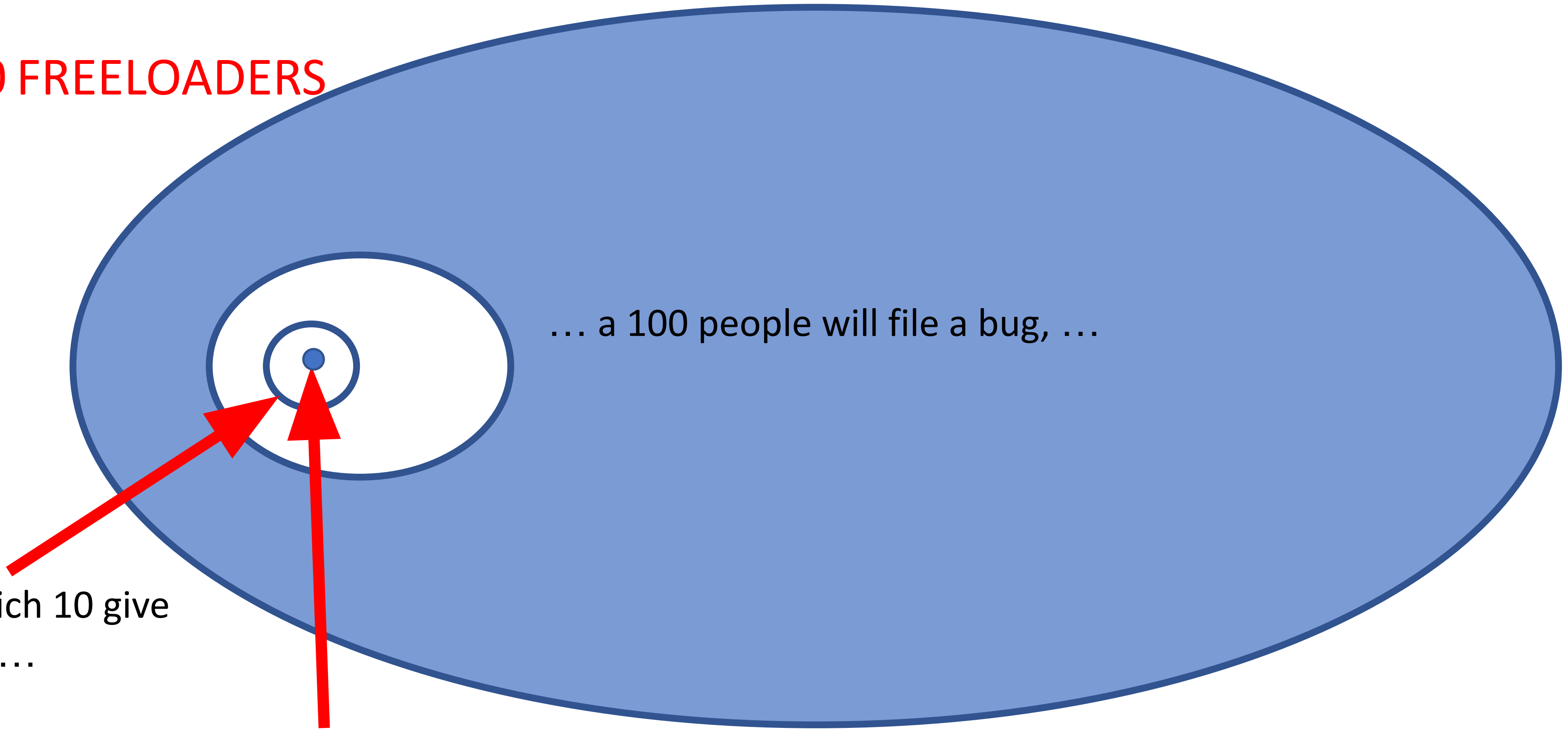
… a 100 will file a bug, …

… out of which 10 give you a patch, …

… out of which 1 actually read your contribution guidelines.

Rules of Thumb and Orders of Magnitude

For every 900 FREELOADERS

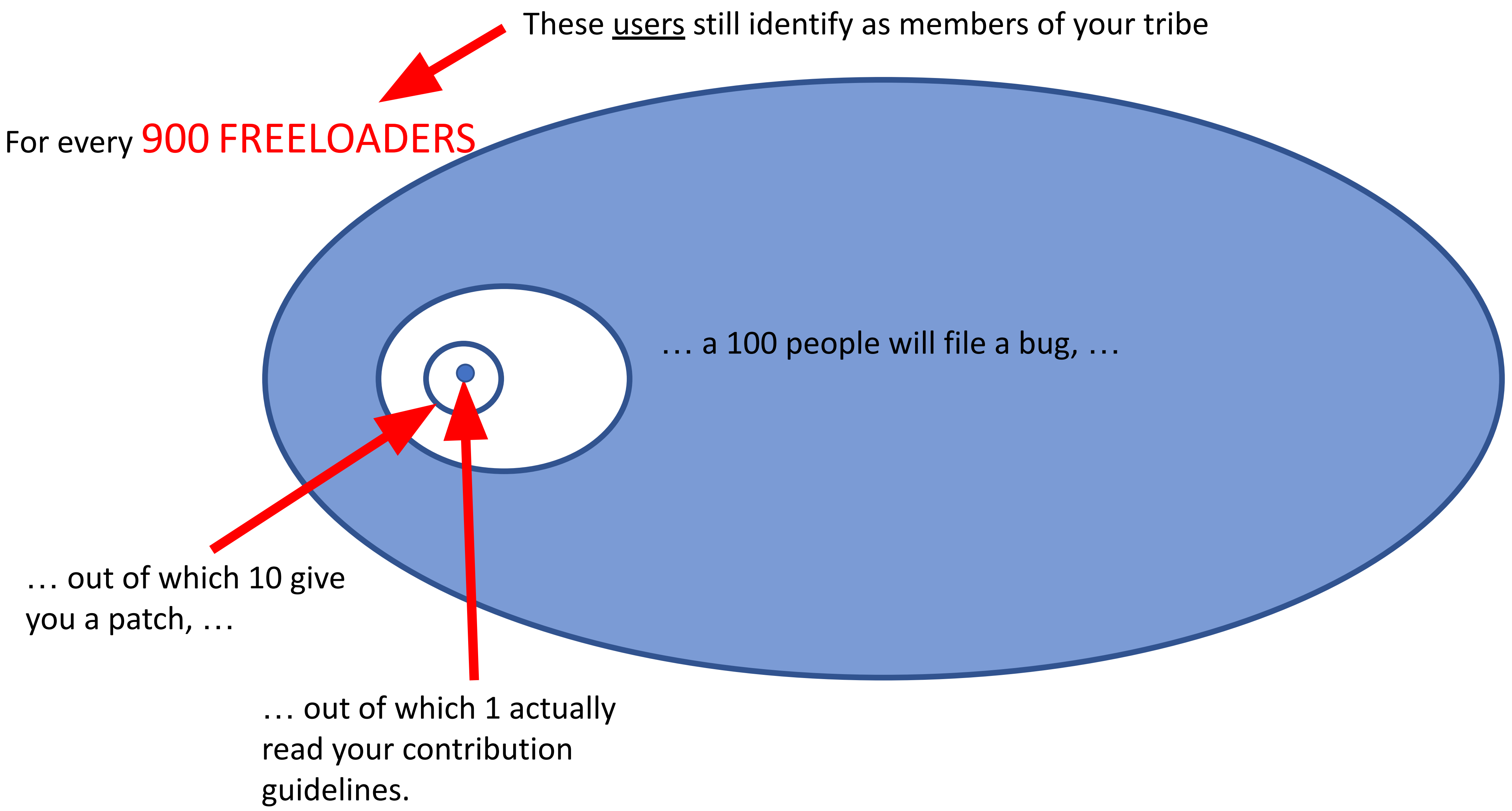


… a 100 people will file a bug, …

… out of which 10 give you a patch, …

… out of which 1 actually read your contribution guidelines.

Rules of Thumb and Orders of Magnitude


These users still identify as members of your tribe
For every 900 FREELOADERS
… a 100 people will file a bug, …
… out of which 10 give you a patch, …
… out of which 1 actually read your contribution guidelines.

Freeloaders means you’re doing it right!

What Activities Make Onboarding Easier?

What are important Community Development Activities for a Project?

What are important Software Construction Tools/Activities for a Project?

Community Development Activities

- The project license is easy to find as this is the outbound statement on how they share.
- There is easy **TESTED** on-boarding documentation (e.g., FAQ, How-to, startup tutorials).
- There is an easy engagement mechanisms (e.g., IRC, email distribution, forums).
- There is a mission statement.
- There is a Code-of-Conduct.
- It is clear which communications channels to use for what purpose.
- There are contribution guidelines.
- The project governance is well documented.
- There are real world events (e.g., conference BoF, Meet-ups).

Software Construction Activities – I

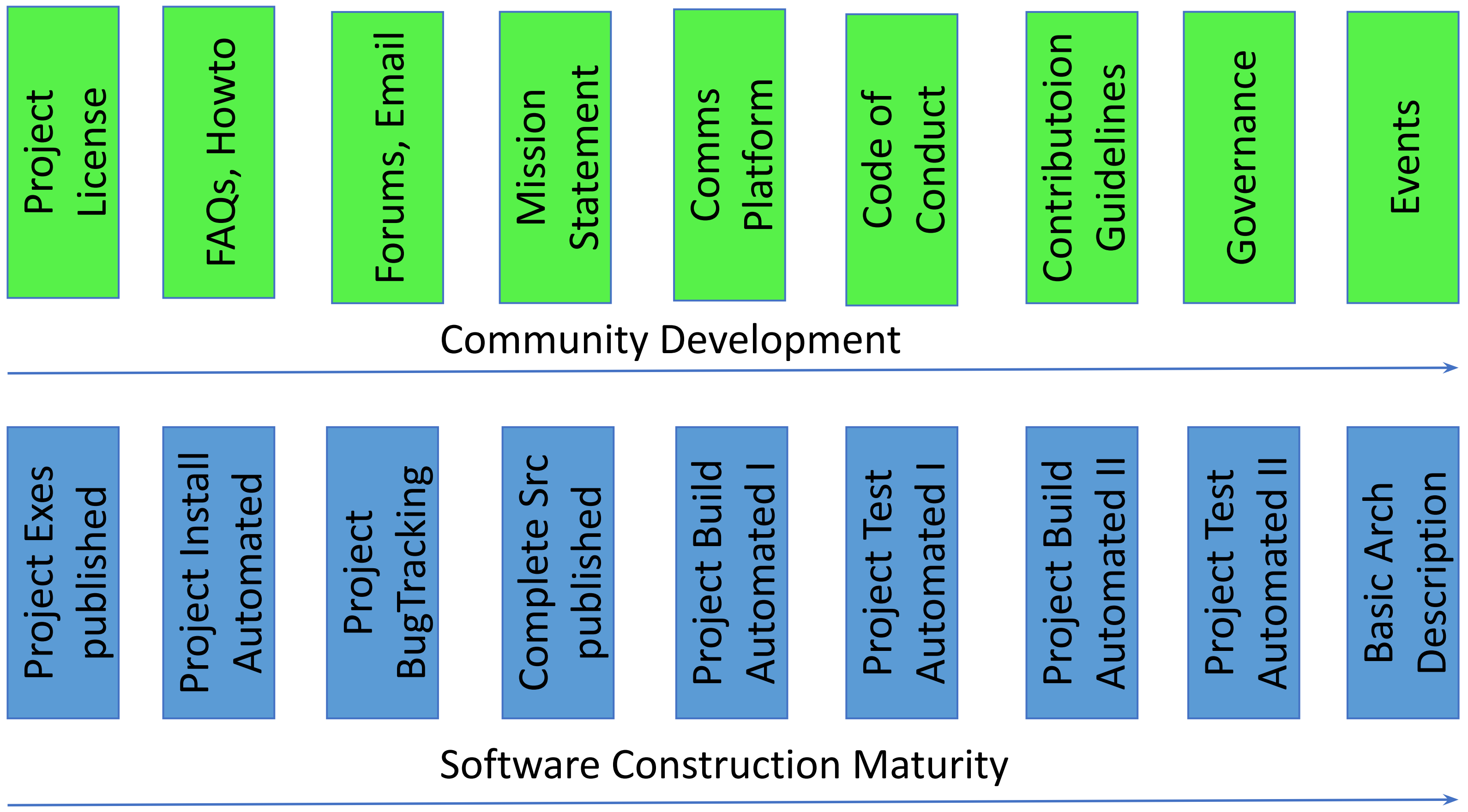
- Consistent executables are built and available on known platforms.
- Project build is automated and instructions **TESTED on all platforms**.
- Project has an automated installer for known platforms.
- Complete source is published and easy to fork/clone/download.
- Software source can be navigated to aid understanding.
- Project can be tested to a known state for known platforms.
- Bugtracking or issue tracking is available.
- Procedures for bug reports, bug triage, new feature requests, contributions (code, docs, tests, etc.) are documented, up-to-date, so everyone engaged has their expectations set.

Software Construction Activities – II

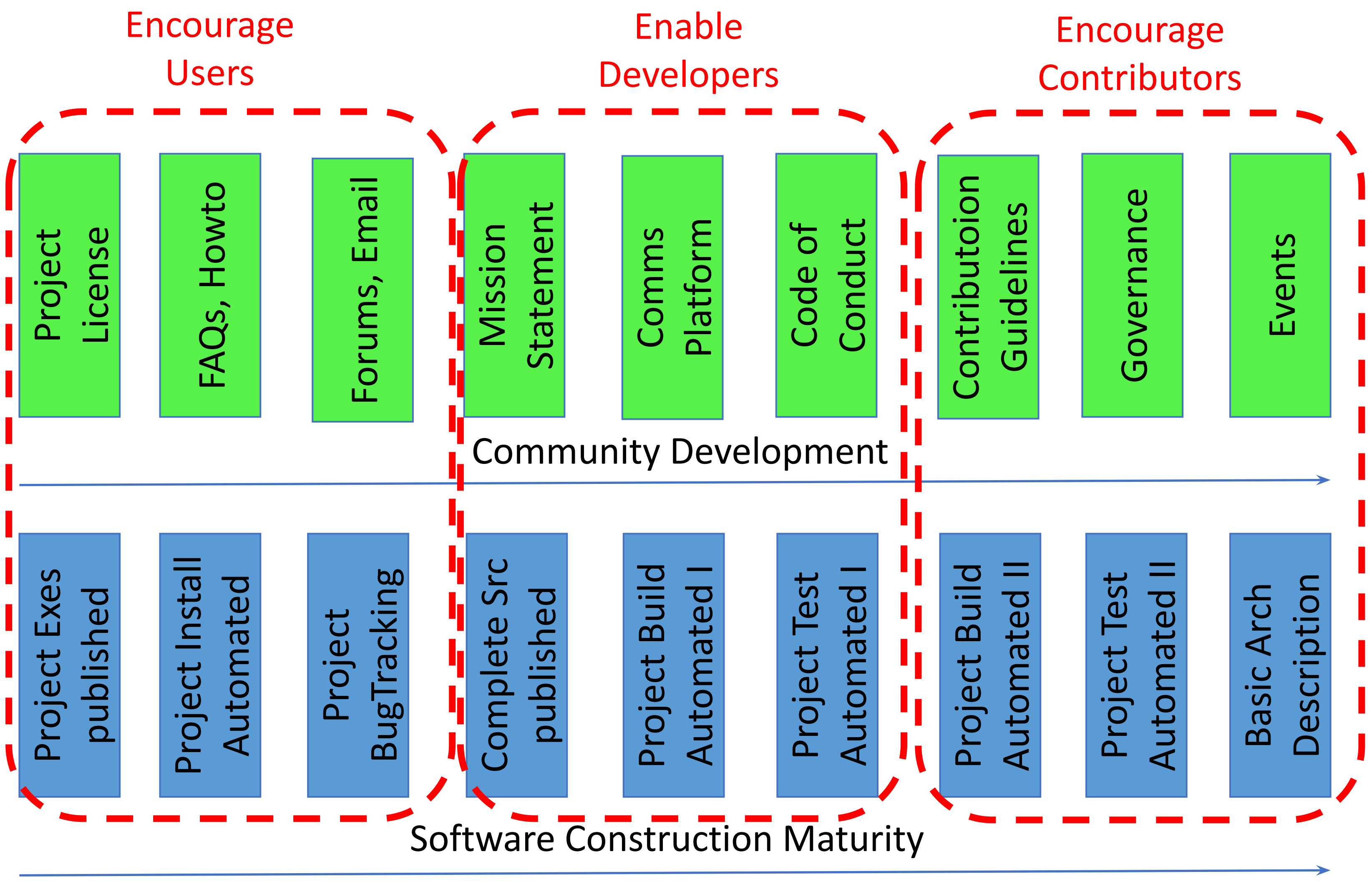
For larger communities with a larger contribution flow:

- Project build pipeline support(e.g., linting, compliance scanning, etc.).
- Sophisticated project testing as part of a build pipeline.
- Architectural documentation exists as well as roadmap discussions, and how roadmap decisions are made is transparent and clear.

Open Source Community Practices



Open Source Community Patterns

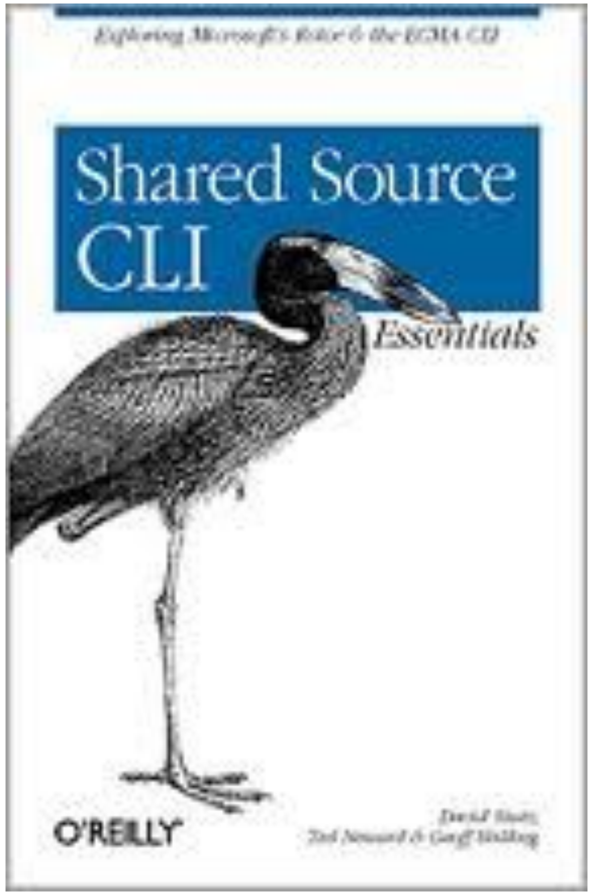


What Does Your 10-Minute Rule Look Like?

What’s your ‘hello world’ scenario?

What’s your mean time to dopamine

ROTOR in 2002 (now known as .NET CORE)


Exploring Microsoft's Rotor & the ECMA CLI
Shared Source CLI Essentials
O'REILLY
David Stutz, Ted Neward & Geoff Shilling

500K LoC
500K Lines-of-Test Harness
Ran on Windows, Mac OS X, FreeBSD
One script to set environment
One command to build everything
One command to test it all
Minimal documentation

24 hours later …
24 hours later again …

There is no community except for the one you build!

Consumers and Producers of OSI-licensed Projects

- As a consumer:
  - How do I use the software? Can I use the project? Where do I ask questions?
  - How do I build the software? Can I use the project? Did I build it <u>correctly</u>?
  - How do contribute? What can I contribute? Where do I contribute?
- As a producer:
  - How do you make it easy to install/configure/use the software?
  - How do you make it easy to build/test/experiment?
  - How do you make it easy to contribute?

‘Community’ Development Activities

Users

- The project OSI-approved license is easy to find as this is the outbound statement on how the project shares its software under copyright law.
- There is easy on-boarding documentation (e.g., FAQ, How-to, startup tutorials).
- There is an easy engagement mechanisms (e.g., IRC, email distribution, forums).

Developers

- There is a mission statement. (Most users won’t care – developers probably do.)
- There is a Code-of-Conduct.
- It is clear which communications channels to use for what purpose.

Contributors

- There are contribution guidelines. (What’s a good contribution? How it happens)
- The project governance is well documented.
- There are real world events (e.g., conference BoF, Meet-ups).

Maturing Community Process

Software Construction Activities – I

Users

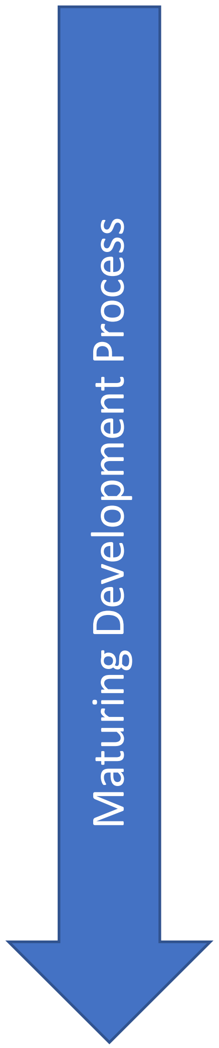
- Consistent executables are built and available on known platforms.
- Project has an automated installer for known platforms.

Developers

- Project build is automated.
- Complete source is published and easy to fork/clone/download.
- Software source can be navigated to aid understanding.
- Project can be tested to a known state for known platforms.

Contributors

- Bugtracking or issue tracking is available.
- Procedures for bug reports, bug triage, new feature requests, contributions (code, docs, tests, etc.) are documented, up-to-date, so everyone engaged has their expectations set.



Additional Software Construction Activities in Very Large Projects!

For larger communities with a larger contribution flow:

- Project build pipeline support(e.g., linting, compliance scanning, etc.).
- Sophisticated project testing as part of a build pipeline.
- Architectural documentation exists as well as roadmap discussions, and how roadmap decisions are made is transparent and clear.

The User On-Ramp

1. Project Website: Note project website URL.
2. Project License: Note project license URL. How many licenses are there?
3. Introductory Documents: Were there getting started docs or tutorials?
4. FAQ: Was there an FAQ?
5. User Installation: Does it look like you can install the project executables without building the project from source?
6. What platforms are supported?
7. Platform Installer Support?
8. Bugtracking/Issues Tracking?
9. How-to: Tutorials?

The Developer On-Ramp

10. **Source Code easy to find?**
11. **Are there good instructions for how to build the project?**
12. **Can you test to a Known State?**
13. Design Documentation?
14. Project Communications: IRC/Slack channel, any email distribution lists, or forums?
15. Is there a mission statement for the project?
16. Is there a code-of-conduct?

The Contributor On-Ramp

17. Could you find getting involved instructions or?
18. Could you find contribution guidelines?
19. Are there conferences or meet-ups?
20. Source Code Base: Use cloc to determine how big the source code base is.

In purely subjective terms, how did you find the whole experience?