THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

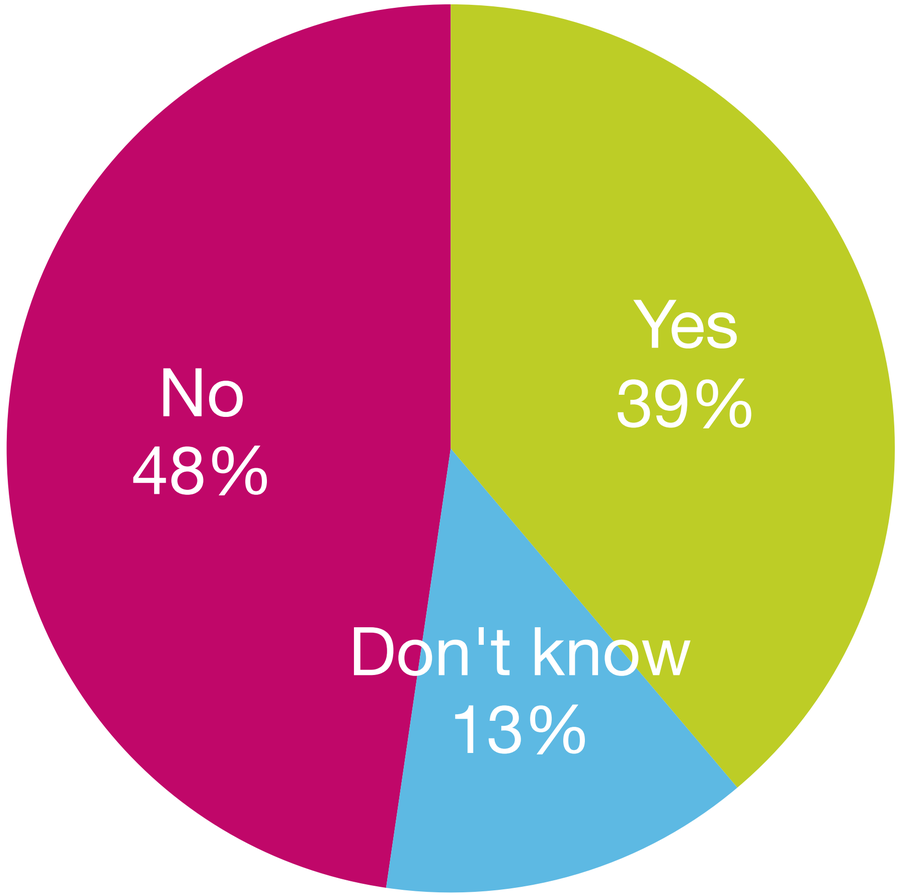
Open source contribution policies that don’t suck!

Tobie Langel, Principal, UnlockOpen

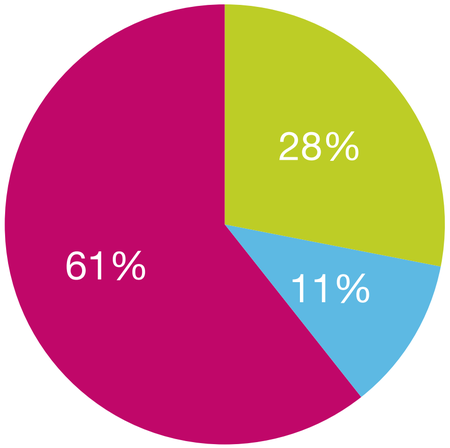
@tobie
tobie@unlockopen.com

THE LINUX FOUNDATION

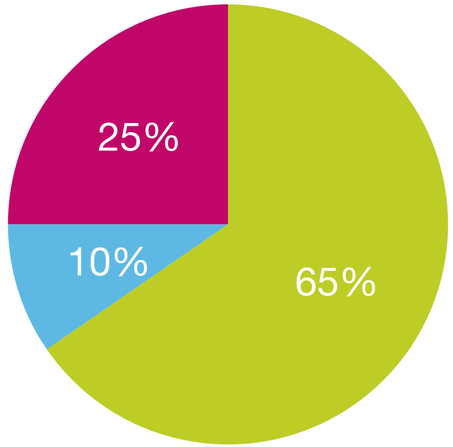
Do you have a open source policy?



< 250 employees



> 10,000 employees



Remember this is a biased sample!

Source: The New Stack & Linux Foundation/TODO Group 2018 Open Source Program Management Survey (https://github.com/todogroup/survey)

What does not having a policy mean?

Contrary to popular belief, it does not mean that you don't have an open source policy at all.

It means that you don't have a *written* one.

You have a policy, whether it is written down or not. It could range from "no open source at all" to "anything goes."

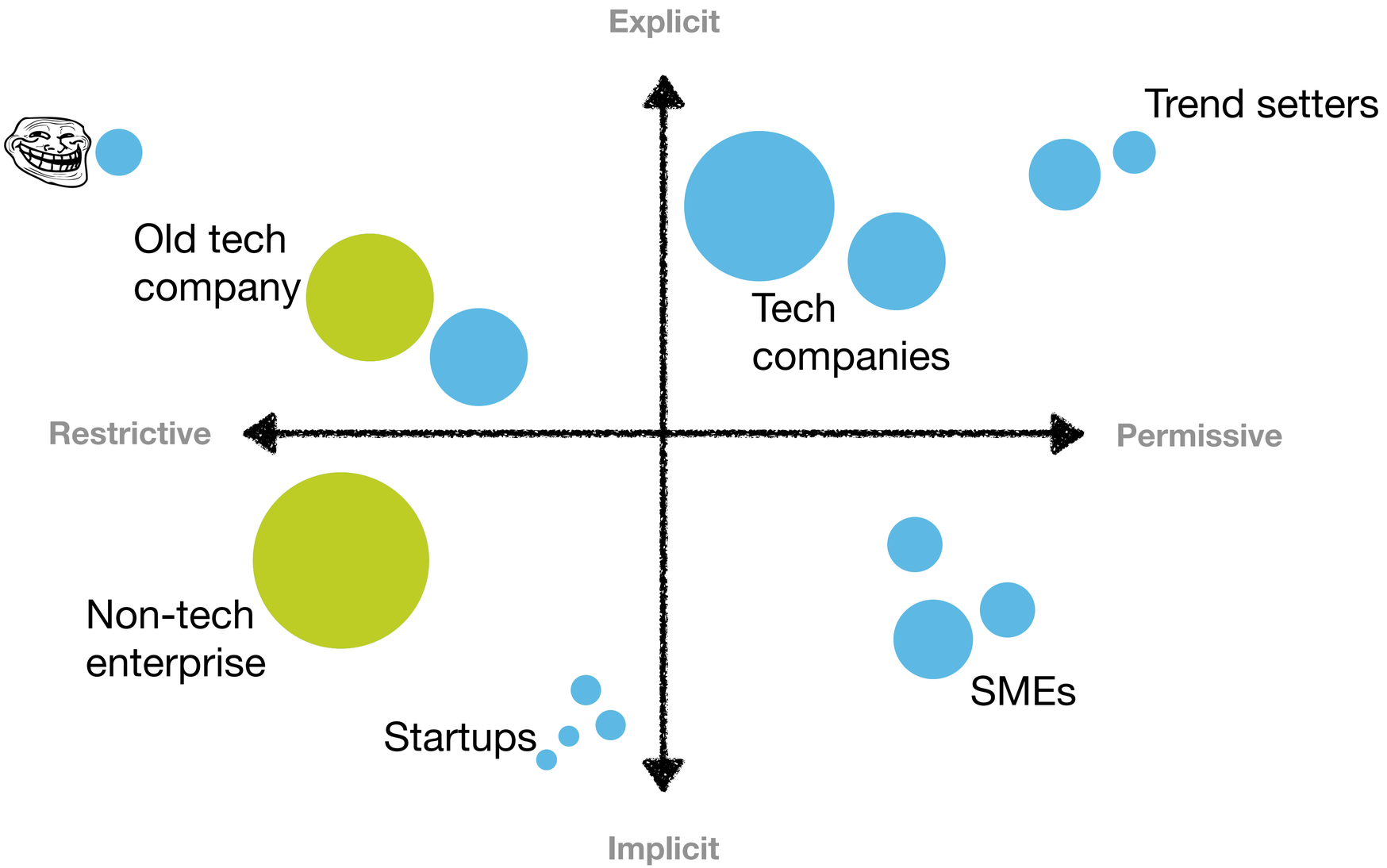
—Heather J. Meeker, Open Source Licensing Specialist,
author of Open Source for Business.

What does having a policy mean?

Think you’re in the right camp because you have an open source contribution policy? Think again!

- You can have a **very restrictive** open source policy.
- You can have a **very bureaucratic** process to obtain approval.
- You can have a **very opaque** process to obtain approval.

None of these are fun!


Explicit
Trend setters
Old tech company
Tech companies
Restrictive
Permissive
Non-tech enterprise
SMEs
Startups
Implicit

What is a policy that doesn’t suck?

Engineering perspective

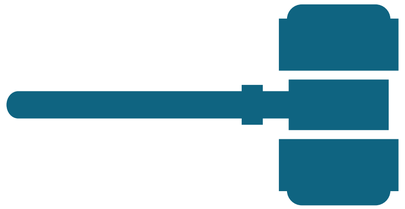
Permissive. Allows open source contribution to be an integral part of the company’s engineering culture and best practices. Based on trust and autonomy.

Explicit. The decision making process is well documented and transparent.

Informative. The policy explains the *why* an helps educate engineers.

Frictionless. Avoid bureaucracy, red tape, lengthly back and forth with legal, etc.

What is a policy that doesn’t suck?



Legal perspective*

Minimizes risk. Avoid:

- giving away competitive advantage,
- giving away IP that can be used defensively (or—*shudders*—offensively),
- reputational damages and accidental infringements.

Consistently followed across the company. Keep contribution under your radar. Avoid compliance issues.

Savvy about written information. Sometimes you want a paper-trail (e.g. compliance), sometimes you don’t.

Doesn’t drown critical problems in a sea of menial issues.

* IANAL (I am not a lawyer). Please come and see me if you are and want to help me improve this slide.

What is a policy that doesn’t suck?

Business perspective

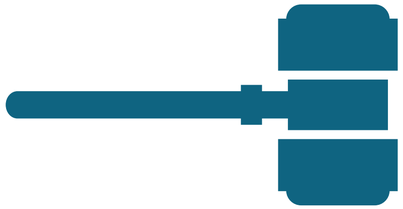
Engineering to be **happy** and **productive**.

Risks minimized and well understood.

Good communication between legal and engineering.

Alignment with Business goals.

At the heart is a tension



Legal wants to **minimize risk**.

Prefers **oral communication**.

Manager’s schedule.

Favors **spectrum thinking**.

Conservative role.

Engineering wants to **maximize velocity**.

Prefers **written communication**.

Maker’s schedule.

Favors **binary thinking**.

Innovative role.

Coming to agreement

Acknowledge that **this tension is normal**. It's just checks and balances.

Listen to both sides.

Remind them that their role is to **help achieve common business goals**.

- Legal's role is to minimize risk, *but not at the expense of innovation*.
- Engineering's needs *can't be fulfilled at the expense of the company's survival*.

Find **common ground**. A good policy will improve the life of both sides.

Align your open source activity **with your business goals.** *If you are a patent troll, then don't do open source.*

Accept that your open source policy **will change *with* your business**.

What is a policy *really* about?

Using
open source

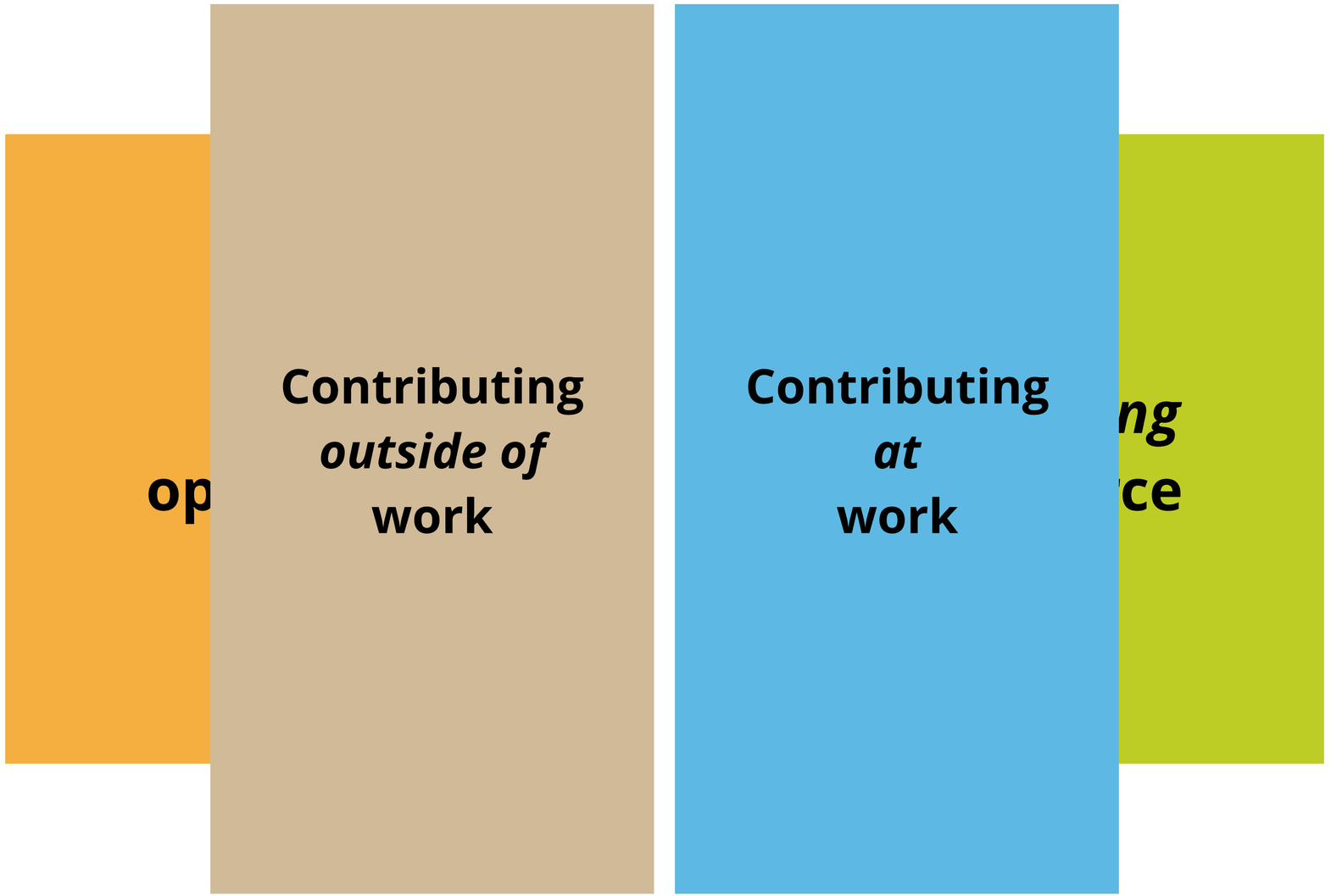
Contributing
open source

Using open source

Well understood problem, essentially:

- Using software with licenses that are compatible with your current *and future* business model.
- Compliance.

Tip: a common issue is missing licenses. Equip engineering with a process (and issue or pull request templates) to request proper OSI-approved licenses.

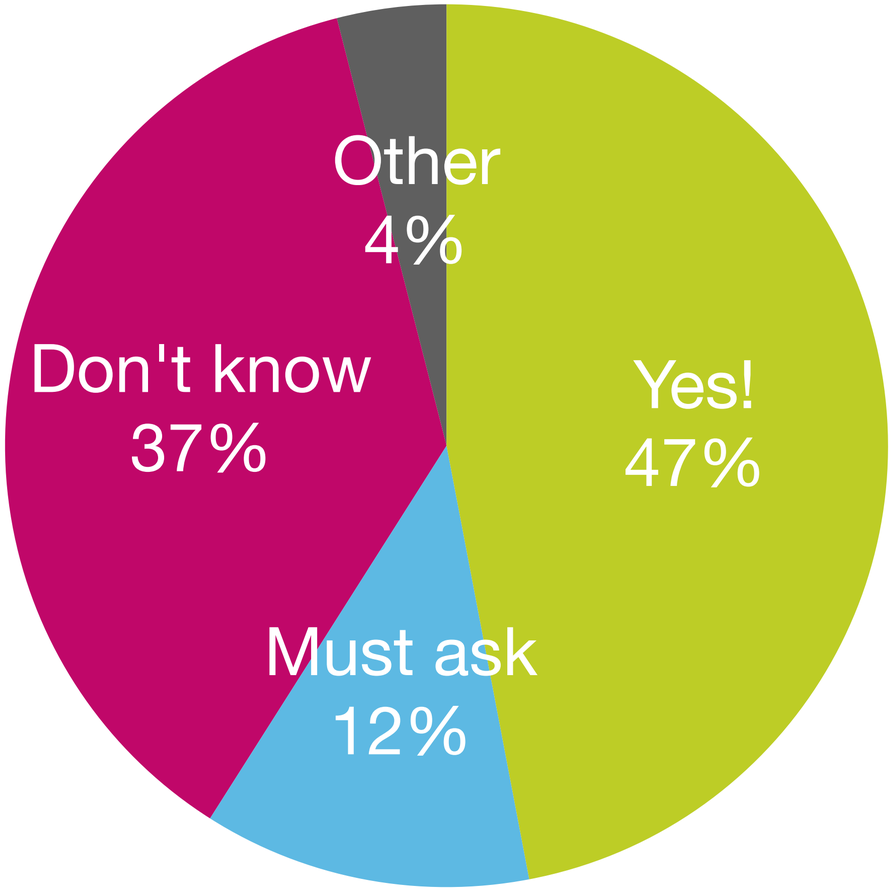

op
Contributing
outside of
work
Contributing
at
work
ng
ce

Can employees contribute to open source on their free time?

- Yes, **without asking** for permission.
- Yes, but **must ask** for permission. *So that sometimes means “no”, right?*
- I don’t know.
- Nope.

Contributing *outside of* work

“How does your employer's IP agreement/policy affect your free-time contributions to open source unrelated to your work?”



But again... this is a highly biased sample.

“Respondents were sampled randomly from traffic and qualifying activity to licensed open source repositories on GitHub.com and invited to complete the survey through a dialog box. A smaller sample was recruited from open source communities sourced outside of GitHub, [...]”

Source: GitHub 2017 open source survey (http://opensourcesurvey.org/2017/).

Why so much confusion?

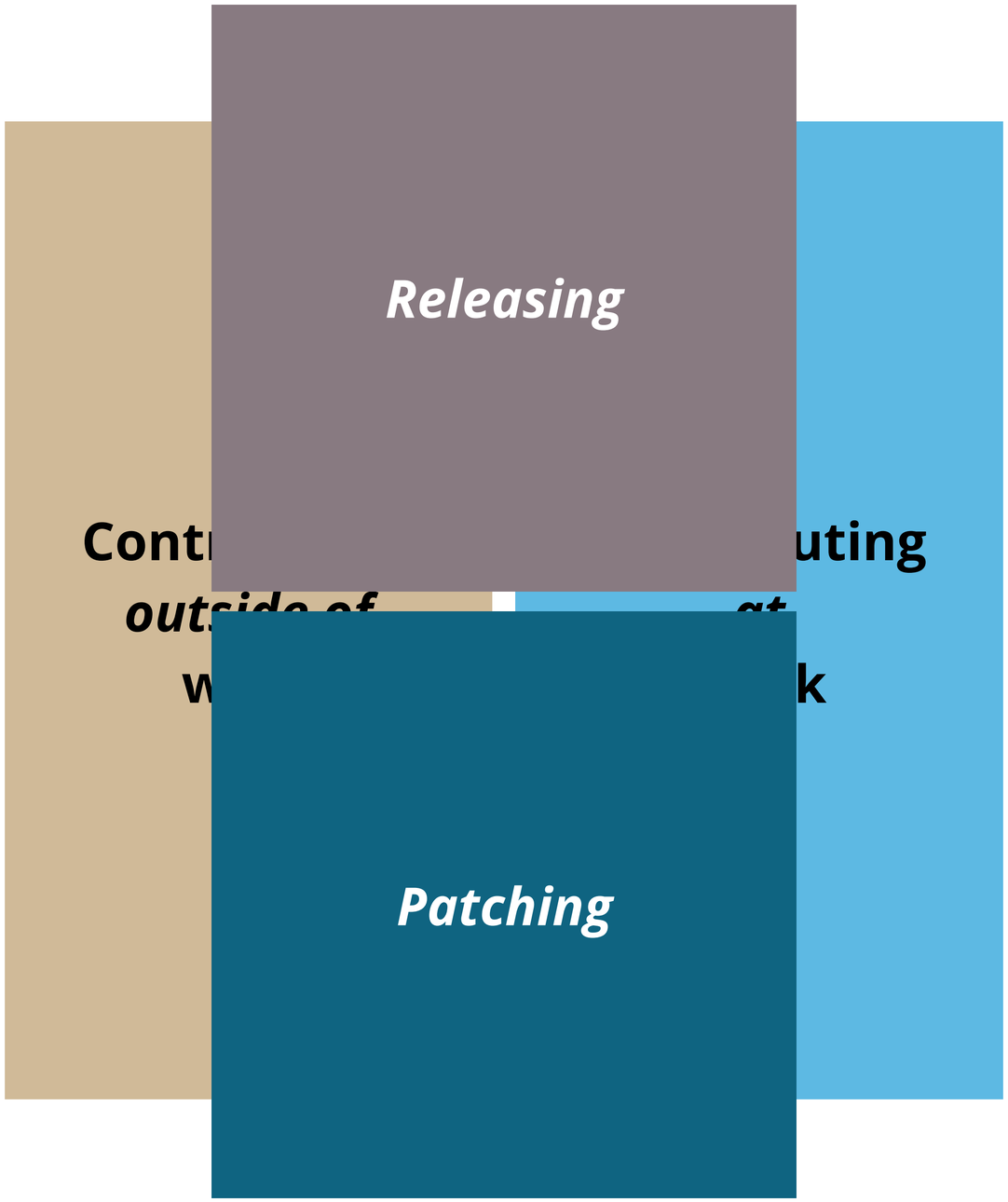
- Depends on the jurisdiction, but not uncommon, especially in the USA, for companies to **own employees' production 24/7**.
- Sometimes, extra criteria apply. For example, in California, IP developed with company equipment—even outside of work—belongs to the employer.
- This **prevents employees from contributing to open source**, unless they ask for, and are granted permission to do so.

The common solution: ask for permission

- Most companies have a process for this.
- Tends to focus on releasing open source or working on a limited set of pre-approved projects.
- Breaks down when there's a high number of dependencies (such as for Node.js projects).

The better solution: BEIPA

- Balanced Employee IP Agreement
- https://github.com/github/balanced-employee-ip-agreement
- Project created by GitHub, based on its own IP agreement.
- BEIPA only claims control of creations made for or relating to the company's business.


Releasing
Patching

Releasing open source at work

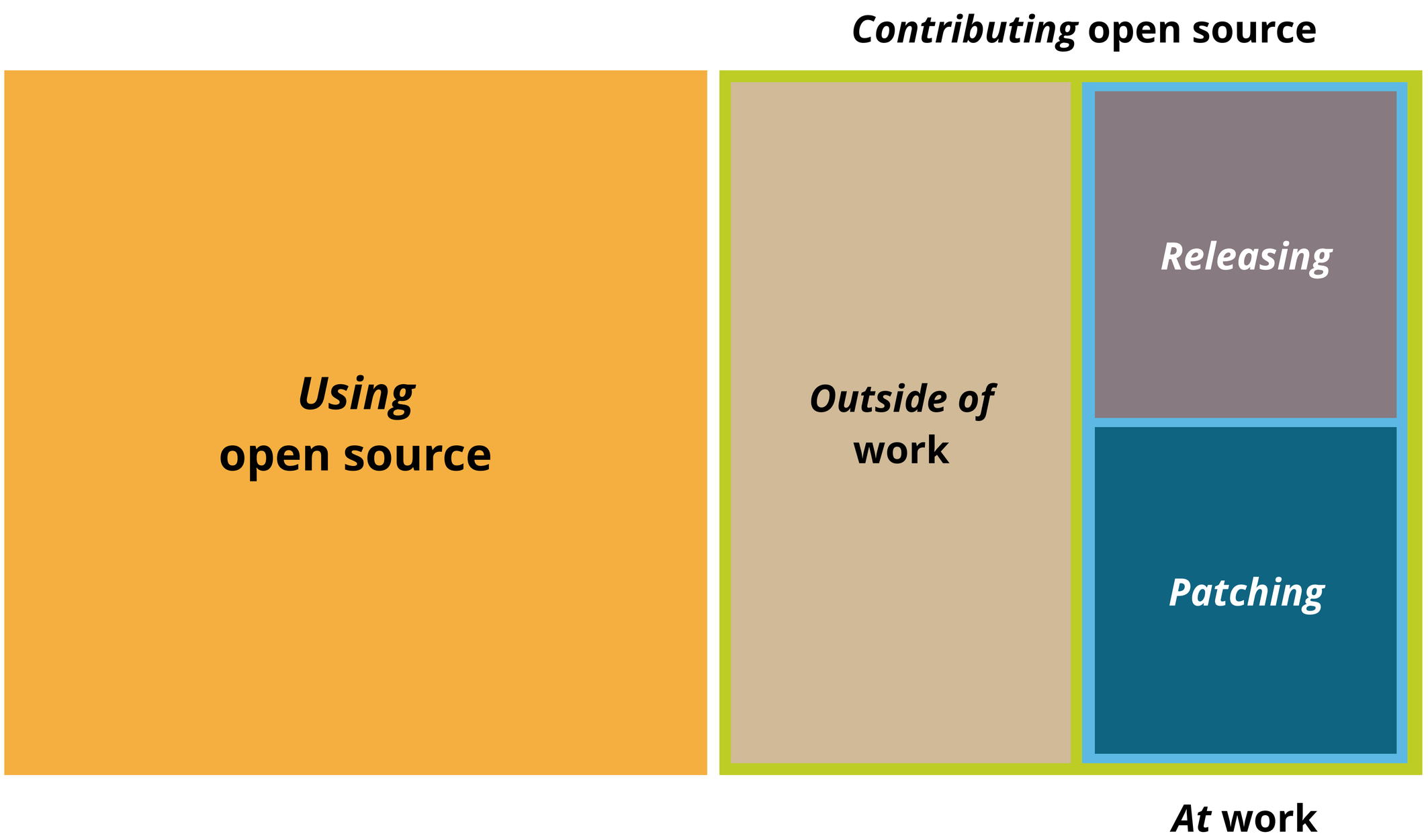
- **Distinguish** large open source projects you want to promote from smaller "day to day" modules. *(E.g. Google's < 100 LoC rule.)*
- Offer **well oiled** and **well documented processes**, checklists, templates, and tooling *(see: https://github.com/todogroup/policies).*
- Offer **help**.
- Promote **working in the open** rather than releasing software once it's done. *(Consider README-driven development to avoid scope creep.)*

Releasing

Patching

Patching open source at work

- By far the **most common activity** and the **most important** one.
- The experience must be as **frictionless** as possible for engineers.
- **Surface the process** by which decisions are made and **trust** engineers to do the right thing. This let's legal **focus on the difficult cases**.
- **Cache** decisions (build approve- and deny- lists) so that the process gets faster as time goes by.


Contributing open source
Using
open source
Outside of
work
Releasing
Patching
At work

Turn your policy into an app!

- **Automatically approve** requests that meet pre-established requirements (e.g. patch an MIT-licensed open source project on GitHub).
- **Automatically reject** requests that don’t meet your criteria (but allow motivated appeals).
- **Manually handle** other requests and cache the decision so more gets automated over time.

Turn your policy into an app!

Using such a system, Adobe was able to **shorten it's review time** from **4.6 days** to **4.6 hours**.

But there's *more*. The data collected can help:

- **understand** your open source activity,
- **promote** it,
- **connect** engineers unknowingly contributing to the same projects,
- etc.

Thank you!

Tobie Langel (@tobie)
Principal, UnlockOpen
tobie@unlockopen.com