FOSSA

TL;DR

- Open source software licenses need to be managed with the same diligence as proprietary ones
- Most organizations start off with management of open source using spreadsheets, which is a manual process that is prone to failure for five reasons:
  1. Scale of the development team or scale at which open source is used
  2. Complications around tracking dependencies
  3. Accuracy of tracking and recency or completeness of licensing data
  4. Change management or workflows around tracking version control of your spreadsheets
  5. Relationships and alignment between legal, security, product, and engineering

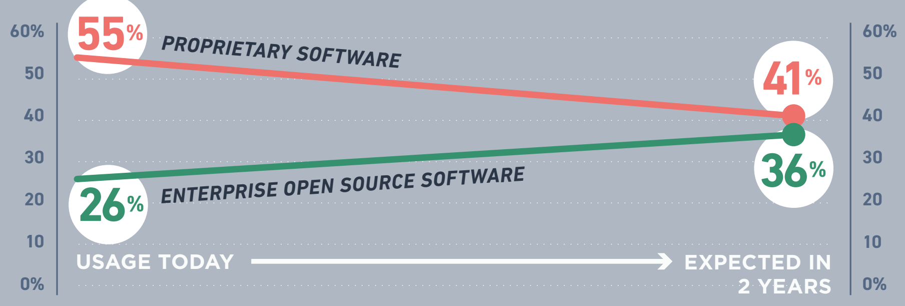
5 Reasons Why Using Spreadsheets for Open Source Management is a Recipe for Disaster

Respecting the terms of the licenses that govern the software created or used by your organization is just as important when the licenses are open source as when they are proprietary. Management of these licenses is an exercise whose importance transcends events like an Initial Public Offering (IPO) or a merger or acquisition (M&A), as revenue could be impacted when potential clients complete their due diligence before a purchase or an online marketplace requires a report before releasing your app on their platform.

AS THE USE OF OPEN SOURCE GROWS AND DEPLOYMENT TIMELINES SHRINK, MANAGEMENT OF OPEN SOURCE IS A GROWING CONCERN THAT MANY ORGANIZATIONS ARE STRUGGLING WITH HANDLING EFFECTIVELY.

Most organizations start off managing all of their open source dependencies in a spreadsheet. This is a process that generally involves someone from legal, engineering, product, or security tracking down the correct engineers to fill out a form that lists every open source component used to help build a product. Then, legal and security review the dependencies against license policy and security vulnerability databases to ensure the software is compliant and secure. Finally, these reports are finalized and shared with auditors for IPOs or Mergers/Acquisitions and/or reports are doled out to customers and partners who require compliance.

2019 TRENDS IN OPEN SOURCE VS. PROPRIETARY SOFTWARE USAGE



Source: "The State Of Enterprise Open Source - Redhat.com." Enterprise Open Source Report 2019, RedHat, 15 Apr. 2019, www.redhat.com/cms/managed-files/rh-enterprise-open-source-ebook-f16984bf-201904-en_1.pdf.

69%

OF ENTERPRISE COMPANIES PLAN ON INCREASING THEIR CONSUMPTION OF OPEN SOURCE SOFTWARE

Sounds like a headache already – but here are five concrete reasons why spreadsheets set you up to fail:

Reason 1: SCALE

Scale destroys this process in two ways: the scale of the team and the scale at which open source software is used.

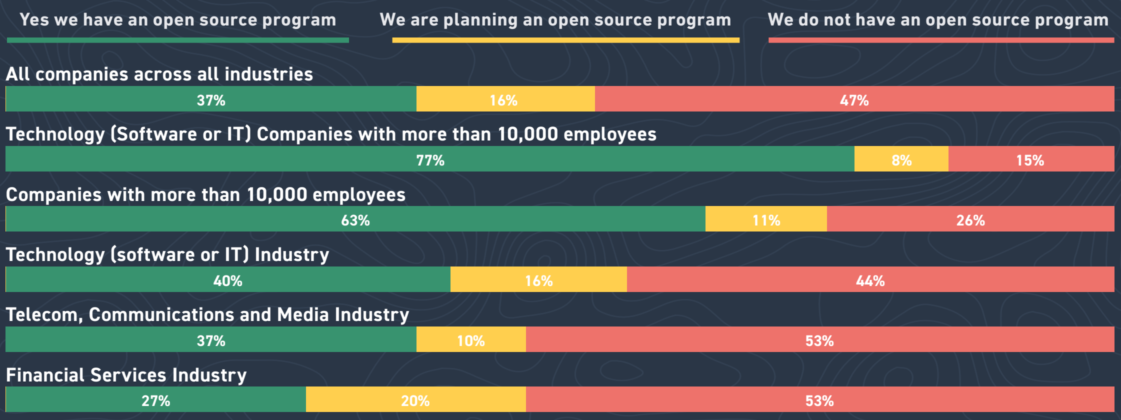
In a growing team, the "spreadsheet approach" can be feasible with 5-10 engineers. At 50 engineers you start to play a game of telephone. Legal has to track down the VP of Engineering, who tracks down the project leader who probably assigns this task to multiple engineers.

What does that look like at 100 engineers? What about 1000? The process is too scattered and decentralized to deliver an accurate report, not to mention the time lost from tracking down the engineers, and the engineers tracking down the dependencies. These productivity hours are much better-leveraged building competitive and innovative products.

The scale of open source software utilization has also drastically changed. Leveraging a spreadsheet was successful when there were only 50 open source projects in existence, but open source has spread at a prolific rate with 72% of companies[1] stating they frequently use open source, 69% of enterprise companies are planning on increasing their consumption of open source software[2]. A spreadsheet simply cannot keep pace.

» INDUSTRY ADOPTION OF OPEN SOURCE

The enterprise has embraced open source, regardless of industry.



Source: Todo Group. "Open Source Programs Survey." *GitHub Todogroup Survey: Open Source Programs Survey*, Todo Group, github.com/todogroup/survey.

Reason 2: Dependencies

Relying on engineers to complete forms and fill out spreadsheets will (at best) track most of the first level dependencies (the open source components they are intentionally using in their projects). However, providing an accurate list of dependencies is not (and shouldn't be) a software engineer's priority. Finding all the dependencies can be difficult, as some projects — such as Uber's open source project Kepler — have more than 1300 dependencies.[3] This type of tracking can easily be automated, while manual tracking consumes valuable engineering resources and takes them away from building the company's products, tools or infrastructure.

The truth of the matter is that any list an engineer compiles will always be incomplete. The direct dependencies you rely on for your product to work also rely on other, external open source components (transitive dependencies). And those transitive dependencies? They also rely on open source components. This tree can go on for quite some time. At the end of the day, your company is responsible for complying with every single license for every single open source component used throughout the dependency tree.

KEPLER, ONE OF UBER'S OPEN SOURCE PROJECTS HAS AROUND 1300 DEPENDENCIES[3]

Reason 3: Accuracy

There's going to be some inaccuracy in your engineers' dependency list if you're tracking manually, but dependency lists aren't the only area with a high likelihood of errors. For example, making sure you have included the correct license with your dependency can be a challenge. Sometimes a license is declared, sometimes licenses are embedded in the actual files, and sometimes a license can be declared and embedded, but the licenses don't match.

Even more challenging is keeping a spreadsheet of all components up to date, especially with modern development practices. If engineers are continuously committing code (CI/CD), your spreadsheet should be updated with every commit. For example, Amazon engineers deploy code every 11.7 seconds.[4] Many, if not most of those commits will include open source packages. It's impossible for anyone to keep up with that.

AMAZON ENGINEERS DEPLOY CODE EVERY 11.7 SECONDS.[4] MANY, IF NOT MOST OF THOSE COMMITS WILL INCLUDE OPEN SOURCE PACKAGES.

3 "Large-Scale WebGL-Powered Geospatial Data Visualization Tool." *Kepler.gl*, kepler.gl/.

4 Novak, Asami. "Most Companies Deploying Code Weekly, Daily, or Hourly." *New Relic Blog*, New Relic, 4 Feb. 2016, blog.newrelic.com/technology/data-culture-survey-results-faster-deployment/.

SICK OF THIS PROCESS? TRY AUTOMATING YOUR WORKFLOW

FOSSA introduces a modern approach to open source compliance so you can focus on bringing a better product to market faster instead of managing your software development through spreadsheets or outdated legacy tools. We give you the tools you need to accelerate product development and collaborate cross-functionally in order to have a competitive product and competitive engineering brand.

Reason 4: Change Management

How can you tell when your spreadsheet is up to date? Do you create a different version? Do you create rules that automatically track when the file is updated? How do you distribute the correct version to the parties involved in listing dependencies? How do you associate the spreadsheet with different deploys or product updates? Keeping your company aligned on what version of a document is the correct version is just a hard problem to solve in general. Doing it when there are so many moving parts? It's an ordeal, to say the least and requires some change management around your open source management, constant diligence, and frequent training (and re-training) as the engineering team grows or changes.

Reason 5: Relationships

This is probably the least quantifiable, but one of the most devastating consequences of relying on manual processes. The relationships between legal, security, product, and engineering can become very strained, very fast. Engineering can feel like legal is breathing down their neck, making them resistant to communication. Even more damaging, engineers will target lawyers as the reason for missed deadlines. Legal can feel isolated because they are responsible for managing risk, but reliant on others with different priorities and deadlines. Product teams can feel exhausted by relaying information to the correct parties on different teams and satisfying none of them. This friction can create silos and strain partnerships across organizations.

ABOUT FOSSA:

FOSSA is the world's first Modern Open Source Management platform. Designed for development and legal teams alike, FOSSA provides component intelligence, continuous compliance, and cross-team collaboration solutions that enable engineering excellence and and accelerate market capture while mitigating business risk.

FOSSA

FOSSA.com | **Sign up with Github** | **tldrlegal.com**

FOSSA, Inc. | Modern Open Source Management
114 Sansome St., Suite #210
San Francisco, CA 94104