CE879 - Information Security Mng. & Eng.

Lecture 9: Open Source & DRM Issues

Seyedeh Atefeh Musavi / Mehdi Kharrazi
Department of Computer Engineering
Sharif University of Technology
Spring 1404

S4Lab

Acknowledgments: Some of the slides are fully or partially obtained from other sources. A reference is noted on the bottom of each slide to acknowledge the full slide or partial slide content.

Agenda

- Philosophical view to open source
- Open source projects ecosystem
- Open source policies for enterprises
- DRM policies

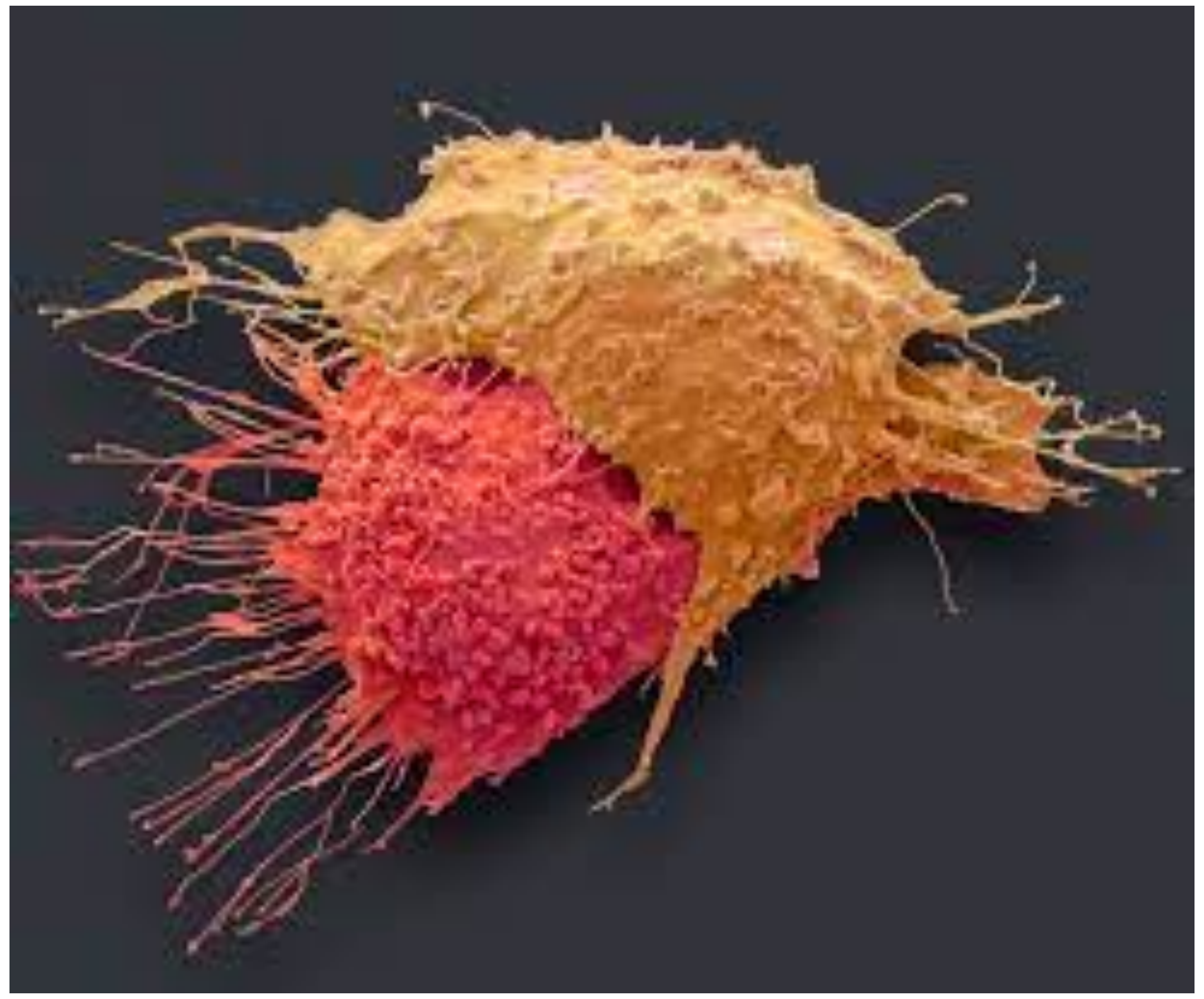
Philosophical view to open source

Incorporating the digital commons: corporate involvement in free and open source software، Birkinbine, B., University of Westminster Press, 2020.

The story starts with a cancer!

[Linux and its GNU GPL license is] “a cancer that attaches itself in an intellectual property sense to everything it touches”

Steve Ballmer, Chief Operating Officer, Microsoft, 2001.



But a bizarre report after!

- In March of 2012, The Linux Foundation released a report entitled, ‘Linux Kernel Development: How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It’.
- The authors included a curious note in the report’s highlights:
    - Microsoft was one of the top 20 contributors to the kernel.
    - But not the only corporation in the top 20.
        - Intel, IBM, Google, Texas Instruments, Cisco, Hewlett-Packard, and Samsung, etc.
- Why, then, would major corporations contribute directly to a FLOSS project, especially when that project seemingly does not directly contribute to corporate profits?

THE LINUX FOUNDATION

Linux Kernel Development

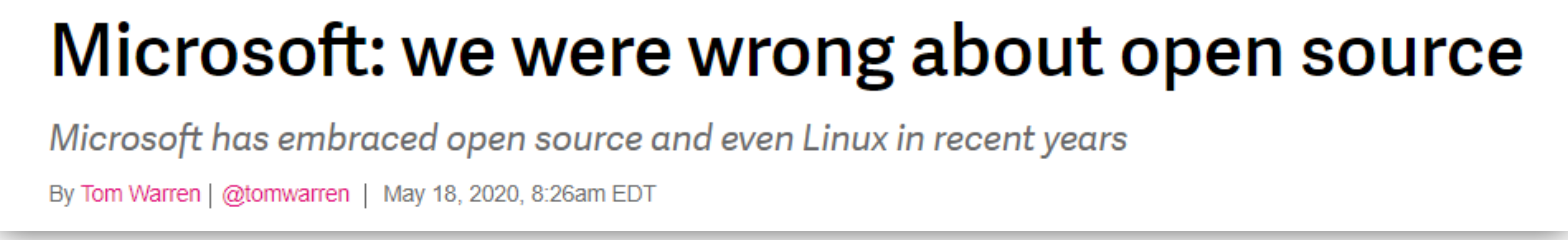
How Fast it is Going, Who is Doing It,
What They are Doing, and Who is Sponsoring I

Jonathan Corbet, LWN.net
Greg Kroah-Hartman, The Linux Foundation
Amanda McPherson, The Linux Foundation

[Incorporating the digital commons: corporate involvement in free and open source software, Birkinbine, B., University of Westminster Press, 2020]

How to analyze FLOSS behaviors of giants

S4Lab



Microsoft: we were wrong about open source

Microsoft has embraced open source and even Linux in recent years

By Tom Warren | @tomwarren | May 18, 2020, 8:26am EDT

- So why Free/Libre and Open Source Software (FLOSS) in giants?
- To follow the reason of this paradoxical behavior! In todays companies we need to realize the FLOSS in depth.
- Such a paradox has its root in the history and philosophy of FLOSS.
- Two different movements have been integrated which are free software and open source software.
- While the former was anti-capitalists, the later was not.
- So integration of the two results in a new philosophical creature.

[Microsoft: we were wrong about open source, Tom Warren, TheVerge, 2020]

Philosophy for free in GNU GPL

- There was a programming language called Unix, the intellectual property rights for which were owned by AT&T.
- One of the programmers working at MIT was Richard Stallman
- When he wanted to work with the Unix programming language outside of officially sanctioned spheres, he was denied access to the code by AT&T.
- In protest, he posted messages to computer-based bulletin boards in 1983 announcing that he was developing a Unix-based language that would be available for free so that others could use the language however they saw fit.
- In 1985, Stallman published ‘The GNU Manifesto’, which outlined the goals of his new project.
- Stallman became the figurehead of the movement against proprietary software.
- He viewed access to source code as a fundamental right, which he wanted others to believe in as well.
- Positioning free software as a moral right.

[Incorporating the digital commons: corporate involvement in free and open source software, Birkinbine, B., University of Westminster Press, 2020]

Philosophy of open-ness

- While Stallman is generally considered to be the figurehead of the free software movement, open source software is generally associated with Linus Torvalds.
- Free software had not yet found a way to coordinate efforts on a larger scale.
- Torvalds wanted to work on kernel development for an open-source operating system.
- Rather than relying on numerous programmers all working independently on such a task, Torvalds released the source code for his project, which he was calling ‘Linux’, a portmanteau of his name, Linus.
- Torvalds suggested that anyone who was interested in contributing to such a project was encouraged to do so, if they released their work back to the community so that others could progressively work toward.
- The rationale was that coordinated efforts reduce the amount of redundant work, which was summed up in the adage ‘with many eyes, all bugs are shallow’, which Eric Raymond refers to as ‘Linus’s Law.

[Incorporating the digital commons: corporate involvement in free and open source software, Birkinbine, B., University of Westminster Press, 2020]

Open source software's dialectic

- Contradictory relationship between FLOSS communities and for-profit corporations.
- Free and open source software is dialectically situated between capital and the commons.
- A virtuous cycle whereby an association of software programmers actively contribute to a community that claims collective ownership over FLOSS projects.
- On the other hand, capital attempts to capture the value being produced by floss communities.
- This is not to say that the goals of the free software commoners and capitalist firms are always antagonistic.
- Commercial sponsorship of FLOSS projects tends to ensures the project's longevity
- However, we also have other examples of these relationships breaking down, particularly when it concerns the unwanted encroachment of capital upon commonly held resources like the digital commons.

[Incorporating the digital commons: corporate involvement in free and open source software, Birkinbine, B., University of Westminster Press, 2020]

FLOSS is the new creature

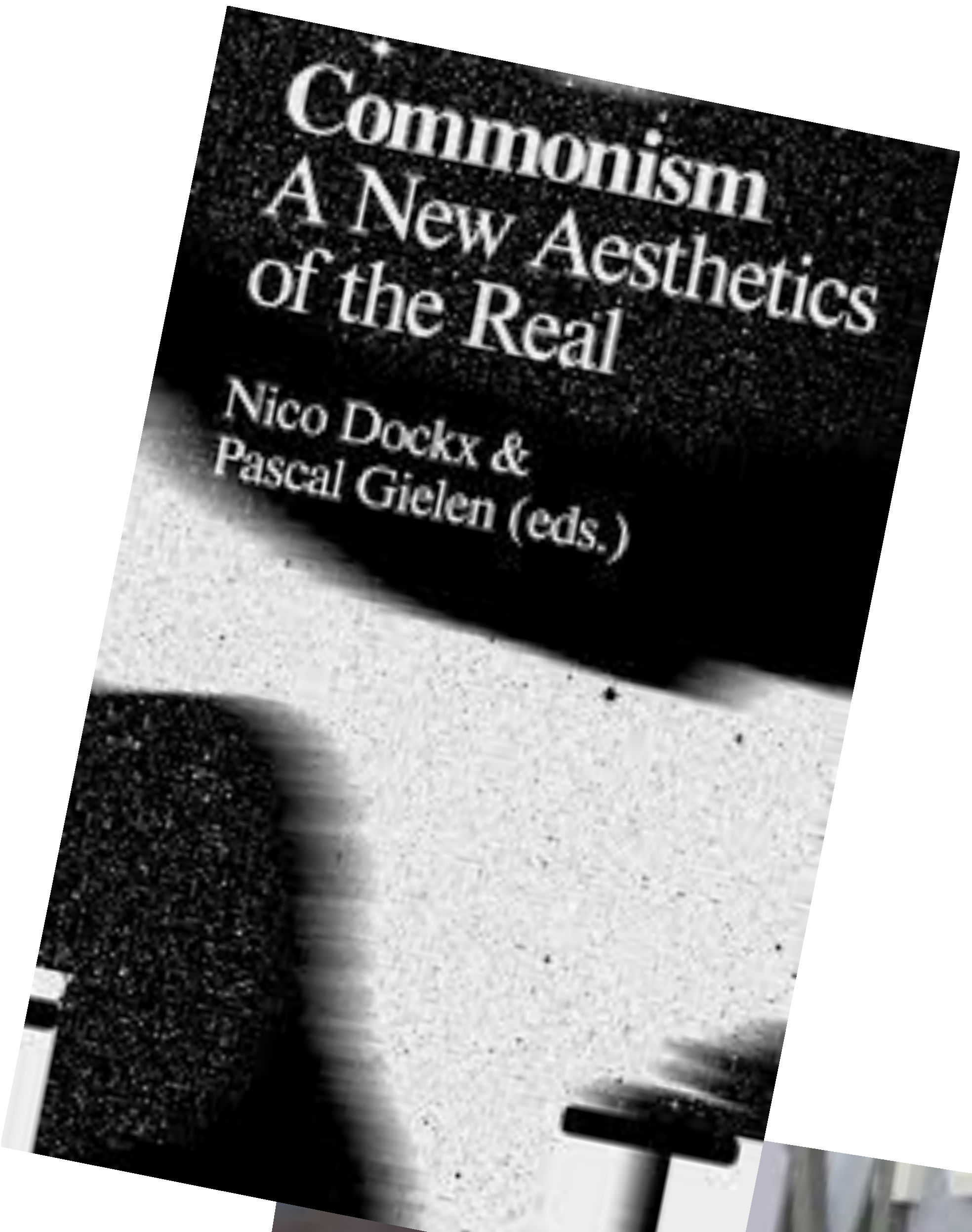
- How, then, to negotiate the relationship between their digital commons and the unwanted intrusion by capital into their projects?
- The commons, generally, and the digital commons, more specifically, can be understood as an alternative system of value that is emerging from within capitalism.
- At times, circuits of commons value can intersect with capital accumulation circuits.
- We will see how this creature is used in today's enterprises

[Incorporating the digital commons: corporate involvement in free and open source software, Birkinbine, B., University of Westminster Press, 2020]

That’s about commonism, not exactly the communism

S4Lab

- After half a century of neoliberalism, a new radical, practice-based ideology is making its way from the margins: commonism.
- It is based on the values of sharing, common (intellectual) ownership and new social co-operations.
- Commoners assert that social relationships can replace money (contract) relationships.
- They advocate solidarity and they trust in peer-to-peer relationships to develop new ways of production.



Commonism
A New Aesthetics of the Real
Nico Dockx & Pascal Gielen (eds.)

[Commonism: a new aesthetics of the real, Dockx, N., & Gielen, P., Valiz, 2018]

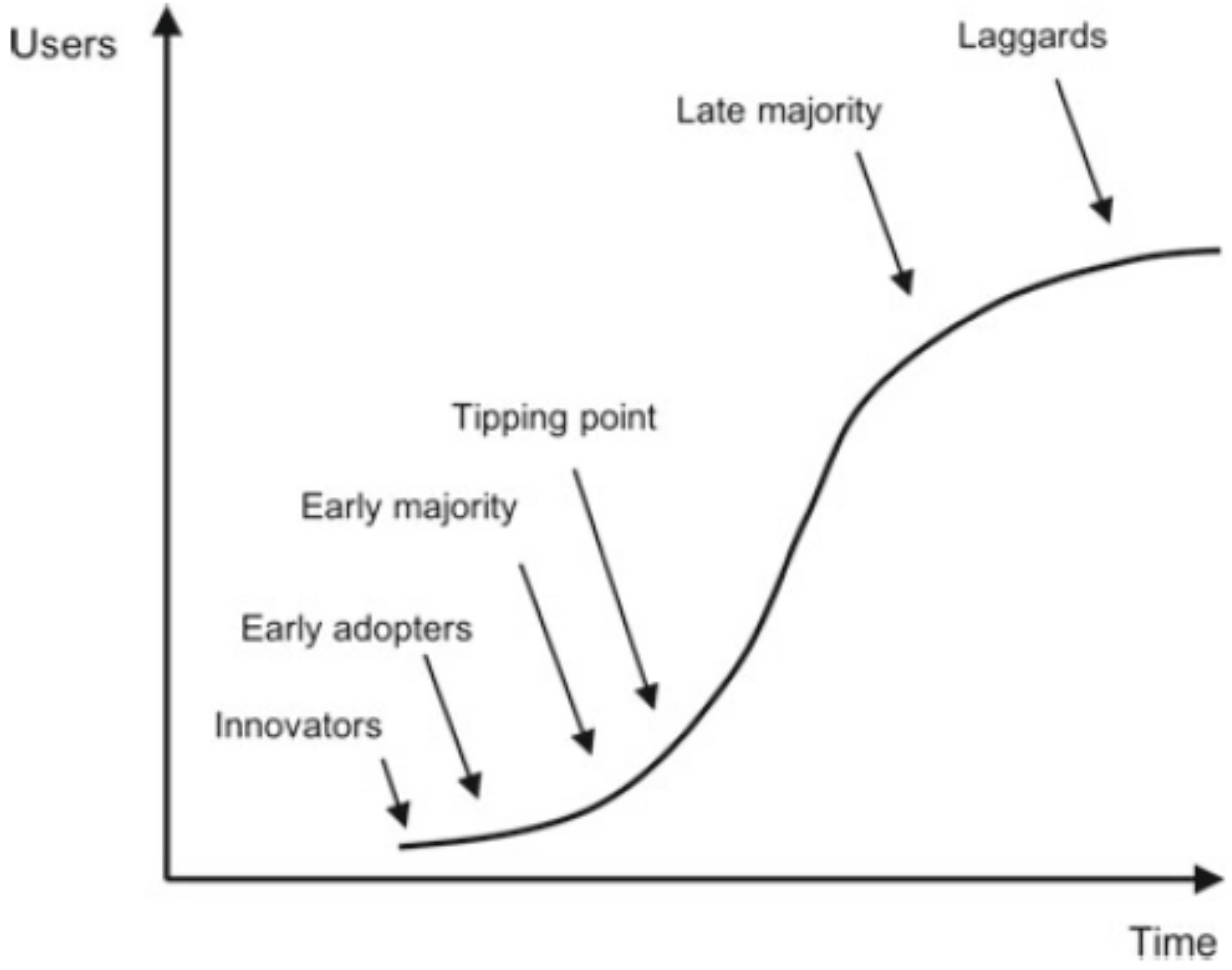
Open source projects ecosystem

A Preliminary Theory for Open-Source Ecosystem Microeconomics, Jullien, N., Stol, K. J., & Herbsleb, J. D., In *Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability*, 2019.

The Three Stages of an Open-Source Project

S4Lab

- Phase 1: The User-Innovator Phase
- Phase 2: Blossoming or Fading
- Phase 3: Maturity and Beyond



[A Preliminary Theory for Open-Source Ecosystem Microeconomics, Jullien, N., et al., In Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability, 2019.]

Three categories of early stage FLOSS projects

- The first category represents the “traditional” FLOSS project, started by one or a few individuals to “scratch an itch”.
  - A key characteristic of this type of FLOSS project is that they are solutions developed by individuals to solve a personal computing problem.
- The second category of FLOSS projects is formerly proprietary software that has been open sourced, such as Netscape’s web browser.
  - The reasons for open sourcing may vary:
    - A company no longer wants to spend resources on maintaining the software.
    - Increase market share, which will also change the business model around the product (e.g., services around the product)
    - A company seeks collaboration in the development of complementary assets
- The third category is that of the so-called “planned” FLOSS projects, typically driven by one or a consortium of companies.
  - One well-known and recent example of this is OpenStack

[A Preliminary Theory for Open-Source Ecosystem Microeconomics, Jullien, N., et al., In Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability, 2019.]

Early-stage FLOSS projects propositions

- Attract developers that perceive the project to be of very high personal value (i.e., It solves a personal problem), and who have low entry barriers to participate (i.e., Highly skilled, strong motivation, sufficient time to participate).
- Projects that offer value beyond "personal interest" will attract a more diverse group of stakeholders than the initial developers.
- The popularity of an early-stage floss project depends on the popularity of the technology the project is written in.

[A Preliminary Theory for Open-Source Ecosystem Microeconomics, Jullien, N., et al., In Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability, 2019.]

Attracting a corporation on phase 1

- Projects that offer considerable potential business value will attract corporate investment if the project's value proposition is compatible with the company's strategy.
- If a software component is not critical for the core business of a company, and has a potential of evolution, the company will favor an open-source strategy to share the cost of development.
- If a software component is critical for the core business of a company, and has a high potential of evolution, an open-source strategy will be considered if and only if the technical structure of the software allows the company to keep some strategic components closed while open-sourcing the standard part to benefit from the innovative dynamic of the community.

[A Preliminary Theory for Open-Source Ecosystem Microeconomics, Jullien, N., et al., In Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability, 2019.]

Phase 2

- As adoption grows, development resources tend to flow into the project, for several distinct reasons.
    - Volunteers are drawn by the increasing visibility and reputation enhancing potential of contributions to the project.
    - Companies are drawn by the high potential, but not yet fully realized, of the project for their business–at this relatively early stage, companies may be able to exert some level of control and shape the project.
- These developers who may have less time and skills than the original core developers, leading to an increased development velocity of the project.
- Sustainable open-source projects are those which succeed in
    1. Structuring their architecture and their organization around modules managed by small teams.
    2. Orchestrating the coordination of the different modules/teams.

[A Preliminary Theory for Open-Source Ecosystem Microeconomics, Jullien, N., et al., In Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability, 2019.]

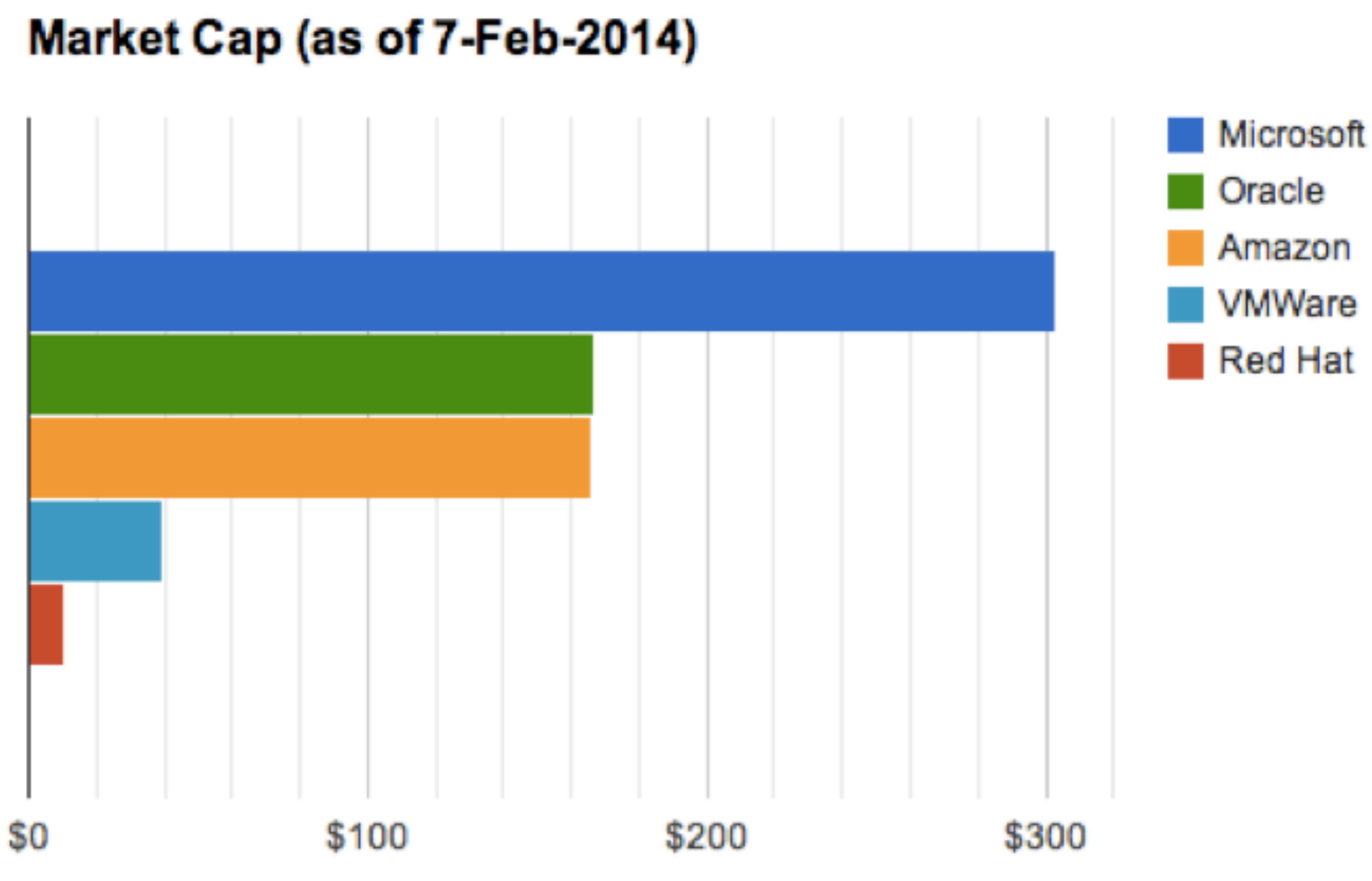
Phase 3

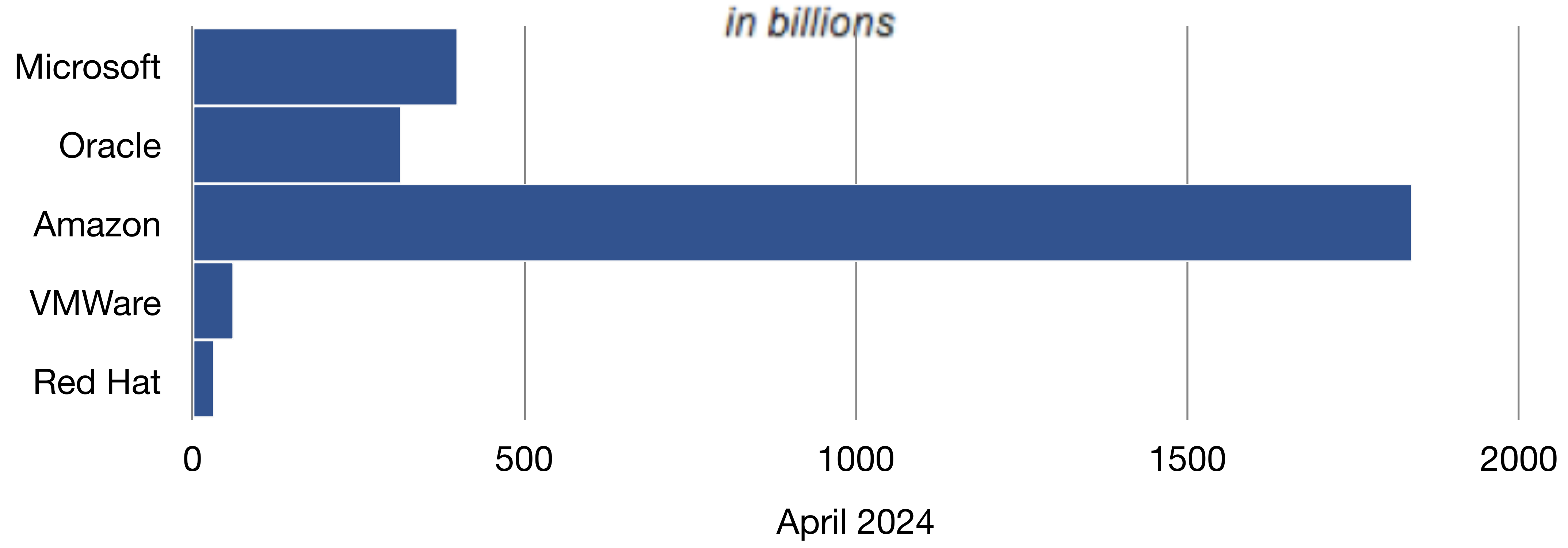
- Projects that are stable in terms of the number of features added/removed will lose developers over time as there is a decreasing amount of work left on the project.
- Mature FLOSS projects tend to become more bureaucratic and rigid in terms of processes and procedures.
- Companies that no longer perceive a project to be of business value will stop investing in that project.
- The continuance of external perturbations leads to continued project activity, even when there is no improvement in terms of functionalities.
- A project's core members are the last to abandon a project (they are the most attached to the project), and the peripheral ones the first.

[A Preliminary Theory for Open-Source Ecosystem Microeconomics, Jullien, N., et al., In Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability, 2019.]

The business model

- The success or failure of open source is not the software itself – it’s definitely up to the tasks required of it – but in the underlying business model.
- We will see four famous business models for open source projects.





[Why There Will Never Be Another RedHat: The Economics Of Open Source, Peter levine, Tech Crunch, 2014]

Open-Core

- In the late 90s and early 2000s, the open-core model was viewed with suspicion.
  - Developers worried that companies building commercial products on-top of open source cores would seek to weaken the open source product to make the commercial offering more attractive.
- Over the last decade this has changed significantly, as we’ve seen countless commercial open-source companies prove to be good stewards of their open-source projects.
- Starting to see design patterns emerge for open-core companies, where their commercial offering complement rather than conflict with the open-core.
  - Ease-of-use pattern: SaaS, UX, Collaboration tools
  - Enterprise pattern: Scalability, Security, Management and Integrations
  - Solutions pattern: Use-case specific functionality

[The Secrets of Successful Open Source Business Models, Imran Ghory, Medium, 2020]

Open-core (2)

- It is important for the community to not feel that essential functionality is being held back from the core product.
  - For most successful open-core companies their customers represent only a small percentage of their overall users.
  - Ensuring the success of the open source product is key to the success of the commercial offering.
- Open-core is now the dominant model for recent open source success stories, including the likes of Confluent, Elastic, and GitLab.

[The Secrets of Successful Open Source Business Models, Imran Ghory, Medium, 2020]

Dual-licensing

- Dual-licensing can be seen as open-core licensing support.
- Usually refers to licensing software under both a proprietary license and an open source license, typically the GPL.
- While this may seem like a conflicting approach, it has become a popular means by which licensors gain the economic benefits associated with commercial licensing while leveraging the community benefits associated with open source licensing.
- It is about distributing the same software under two different license forms:
  - A version subject to a proprietary license (which may come with the right to further develop and commercially distribute that software and with licensor technical support and added features).
  - A version licensed under, and subject to, the restrictions and obligations of an open source license, such as the GPL.

[What is dual licensing? 3 software licensing models to consider, Matt Jacobs, Synopsys, 2017]

snyk	Copyleft					Permissive			
	GPLv3 Free Software Free as in Freedom	AGPLv3 Free Software Free as in Freedom	LGPLv3 Free Software Free as in Freedom	EPL 1.0	MPL	Apache	MIT	BSD	Unlicense
Permissions in addition to commercial use, distribution, modification:									
Patent use	🟢	🟢	🟢	🟢	🟢	🟢	🔴	🔴	🔴
Patent use	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢
Conditions									
Disclose source	🟢	🟢	🟢	🟢	🟢	🔴	🔴	🔴	🔴
License & copyright notice	🟢	🟢	🟢	🟢	🟢	🟢	🟢	Source	🔴
Network use is distribution	🔴	🟢	🔴	🔴	🔴	🔴	🔴	🔴	🔴
Same license	🟢	🟢	Library	🟢	File	🔴	🔴	🔴	🔴
State changes	🟢	🟢	🟢	Some	🔴	🟢	🔴	🔴	🔴
Limitations/Disclaimers									
Liability	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢
Warranty	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢
Trademark use	No explicit limitation				🟢	🟢	🔴	🔴	🔴

Dual license example

- Oracle’s MySQL database management system.
- MySQL under a proprietary (OEM style) license for licensees who want to create and commercially distribute proprietary derivative works incorporating MySQL
- MySQL under the GPL for licensees who simply want to use the software or who want to incorporate MySQL into a product to be later distributed likewise under the GPL.
- Artifex Software, Inc. (“Artifex”) and Hancom controversy.

[What is dual licensing? 3 software licensing models to consider, Matt Jacobs, Synopsys, 2017]

Professional Services (ProServ)

- Early open-source models often built on professional services, with companies paying for support and consultancy.
  - While a number of companies have got to scale with this model, it has not been without significant challenges.
- The margins with professional services are also much thinner than those for product-based companies.
- Services revenue is often highly-unpredictable and requires significant scaling of head-count which can leaves companies exposed when revenues shift.
- While there are a few companies like Hortonworks (pre-Cloudera acquisition) that still have significant revenue via the pure-support subscription model, support is now something that's typically bundled with additional product offerings due to the lack of defensibility.

[The Secrets of Successful Open Source Business Models, Imran Ghory, Medium, 2020]

Hosting

- In the last decade hosting has become a common offering from open source companies, especially in the data space.
- Enabling end-users to use infrastructure components in a similar way to SaaS offerings without having to be concerned with the operational overhead of managing the infrastructure.
- While public open-source companies have generally avoided disclosing margins explicitly for their hosted services, it is estimated to be ~40-65%.
- This places it above services margin but lower than the typical product margin.

[The Secrets of Successful Open Source Business Models, Imran Ghory, Medium, 2020]

Hosting(2)

- In recent years cloud hosting providers, most notably AWS have started to offer managed hosting solutions for common open source packages, further squeezing the margins for open source companies.
- This resulted in a number of open-source products such as Redis and MongoDB changing their licenses to prevent such competition from cloud vendors.
- The other strategy, chosen by companies such Confluent and Elastic, has been to exclusively offer features in their hosted services that aren't available in the open-core, creating a blended hosting/open-core model.
- It's also worth noting that while hosting is often a significant ($100m+) revenue stream for these businesses, it's often the secondary revenue generator rather than the primary.

[The Secrets of Successful Open Source Business Models, Imran Ghory, Medium, 2020]

MongoDB Example

- Some cloud providers are taking MongoDB open-source code and offering a hosted commercial version of its database to their users without playing by the open-source rules.
- MongoDB issued a new software license, the Server Side Public License (SSPL), that will apply to all new releases of its MongoDB Community Server, as well as all patch fixes for prior versions.
- For virtually all regular users who are currently using the community server, nothing changes because the changes to the license don't apply to them.
- MongoDB was previously licensed under the GNU AGPLv3, meaning companies who wanted to run MongoDB as a publicly available service had to open source their software or obtain a commercial license from MongoDB.
- SSPL isn't all that different from the GNU GPLv3, with all the usual freedoms to use, modify and redistribute the code (and virtually the same language), the SSPL explicitly states that anybody who wants to offer MongoDB as a service — or really any other software that uses this license — needs to either get a commercial license or open source the service to give back the community.

[TechCrunch]

Marketplaces

- While often overlooked, the largest commercial success story in open source is Android.
- Mozilla similarly generates the bulk of their $500m annual revenue by providing lead generation to search engines.
  - 2014 annual deal for $375 million to make Yahoo the default search engine in Firefox
- While it's still a relatively rare model, being an intermediary between different parties that interact with your product is a model that open source startups are increasingly exploring, and we're likely to see a number of additional open source companies built on this model over the next decade.

[The Secrets of Successful Open Source Business Models, Imran Ghory, Medium, 2020]

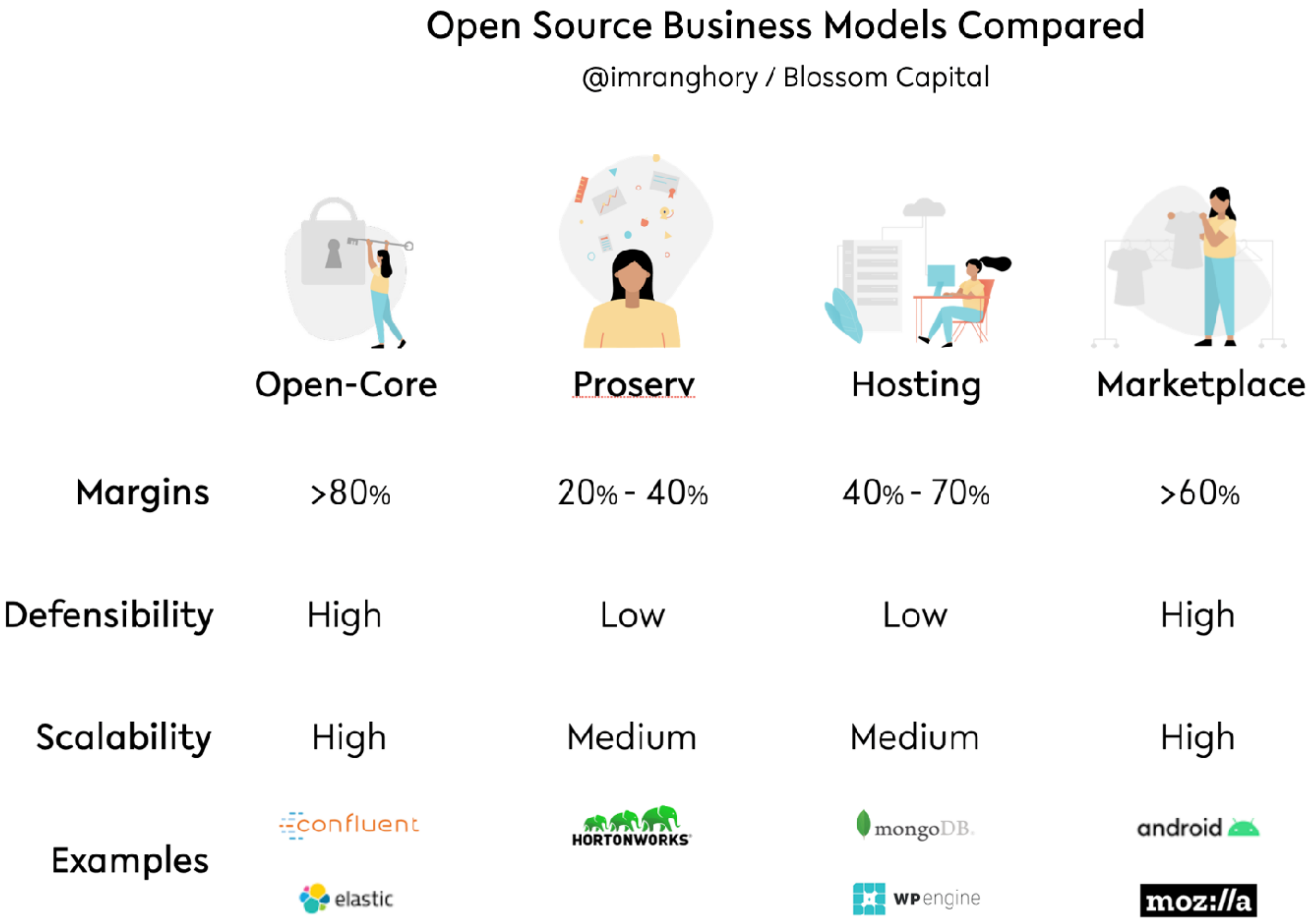
What’s left?

- There is some thing else which is more consistent with our first impression of non-profit view on open source project.
- What’s it?

What’s left?

- There is some thing else which is more consistent with our first impression of non-profit view on open source project.
- What’s it?
- Absent from this list is the “donation-ware," or "pay-what-you-want," model for generating revenue. No large open source company has successfully survived solely on donations.

[4 successful open source business models to consider, Daniel Rubinstein, OpenSource, 2017]



Open Source Business Models Compared

@imranghory / Blossom Capital

	Open-Core	Proserv	Hosting	Marketplace
Margins	>80%	20% - 40%	40% - 70%	>60%
Defensibility	High	Low	Low	High
Scalability	High	Medium	Medium	High
Examples	confluent elastic	HORTONWORKS	mongoDB WP engine	android moz://a

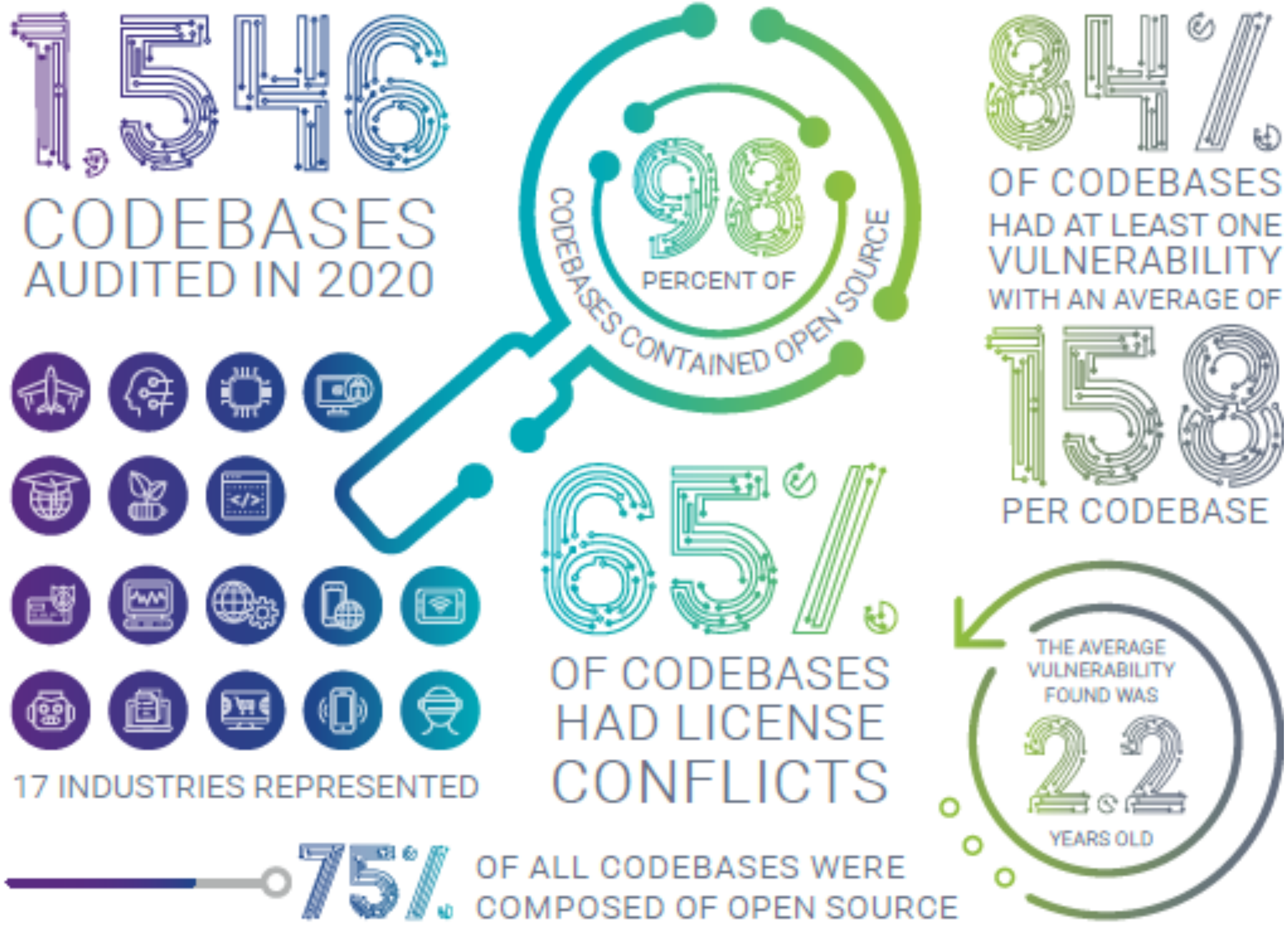
[The Secrets of Successful Open Source Business Models, Imran Ghory, Medium, 2020]

Combinational or new models

- In some cases none of these models might be suitable, and you might need to innovate a unique commercial model for what you are building. You shouldn’t be afraid the explore alternatives, and use the above as a framework for evaluating how well a model may scale.

[The Secrets of Successful Open Source Business Models, Imran Ghory, Medium, 2020]

Open source policies for the enterprise



1,546 CODEBASES AUDITED IN 2020

17 INDUSTRIES REPRESENTED

98 PERCENT OF CODEBASES CONTAINED OPEN SOURCE

84% OF CODEBASES HAD AT LEAST ONE VULNERABILITY WITH AN AVERAGE OF 158 PER CODEBASE

65% OF CODEBASES HAD LICENSE CONFLICTS

THE AVERAGE VULNERABILITY FOUND WAS 2.2 YEARS OLD

75% OF ALL CODEBASES WERE COMPOSED OF OPEN SOURCE

[Open Source Security and Risk Analysis Report, Synopsys, 2021]

Why Are Companies and Governments Turning to Open Source?

- Multiplying the company's investment
  - "The smartest people in every field are never in your own company."
  - At best, an ecosystem of innovation will grow up around an open project. E.g. GeoNode project study.
- Benefiting from the most recent advances.
  - Your data scientists will want implementations of the best and most up-to-date algorithms, and these implementations will usually be open source.
- Spreading knowledge of the software
  - When the code is open—and especially when a robust community grows up around it—adoption is broader.
  - More people throughout the industry understanding the code and the contribution process.

[Open Source in the Enterprise, Andy Oram and Zaheda Bhorat, O'reilly, 2018]

Why Are Companies and Governments Turning to Open Source? (con't)

- Increasing the developer base.
  - A larger pool of talented developers from which the company can hire to work on the code and related projects.
- Upgrading internal developer skills.
  - Developers already recognize that the best way to learn good coding skills is to work on an open source project because they can study the practices of the top coders in the field.
- Building reputation
  - Most people want to work for organizations they can boast about.
- Faster startup of new companies and projects.
  - Working with a community, both on existing software and on your own innovations, saves you time and lets you focus limited employee time on critical competitive parts of your product.

[Open Source in the Enterprise, Andy Oram and Zaheda Bhorat, O'reilly, 2018]

OSS at nation-wide regulations

- Many governments have launched major open source policies and initiatives.
- Some have committed to an “open source first” strategy.
  - Requiring vendors as well as internal developers to use open source licenses and practices wherever possible.
- Governments are realizing that each agency’s needs are similar to other agencies, around the nation and around the world.
- The investment made by one agency can save money for all the rest.
- Open source collaboration also opens opportunities for smaller companies, citizen developers, and nonprofits to contribute to innovation in government services.
- Finally, the software creates a common standard that fosters interoperability for many kinds of development.

[Open Source in the Enterprise, Andy Oram and Zaheda Bhorat, O'reilly, 2018]

What about enterprise secrets?

- Trade secrets and confidential business plans can coexist with open source engagement.
- If even the US national security agency and UK government communications headquarters can use open source software, you can, too.

[Open Source in the Enterprise, Andy Oram and Zaheda Bhorat, O'reilly, 2018]

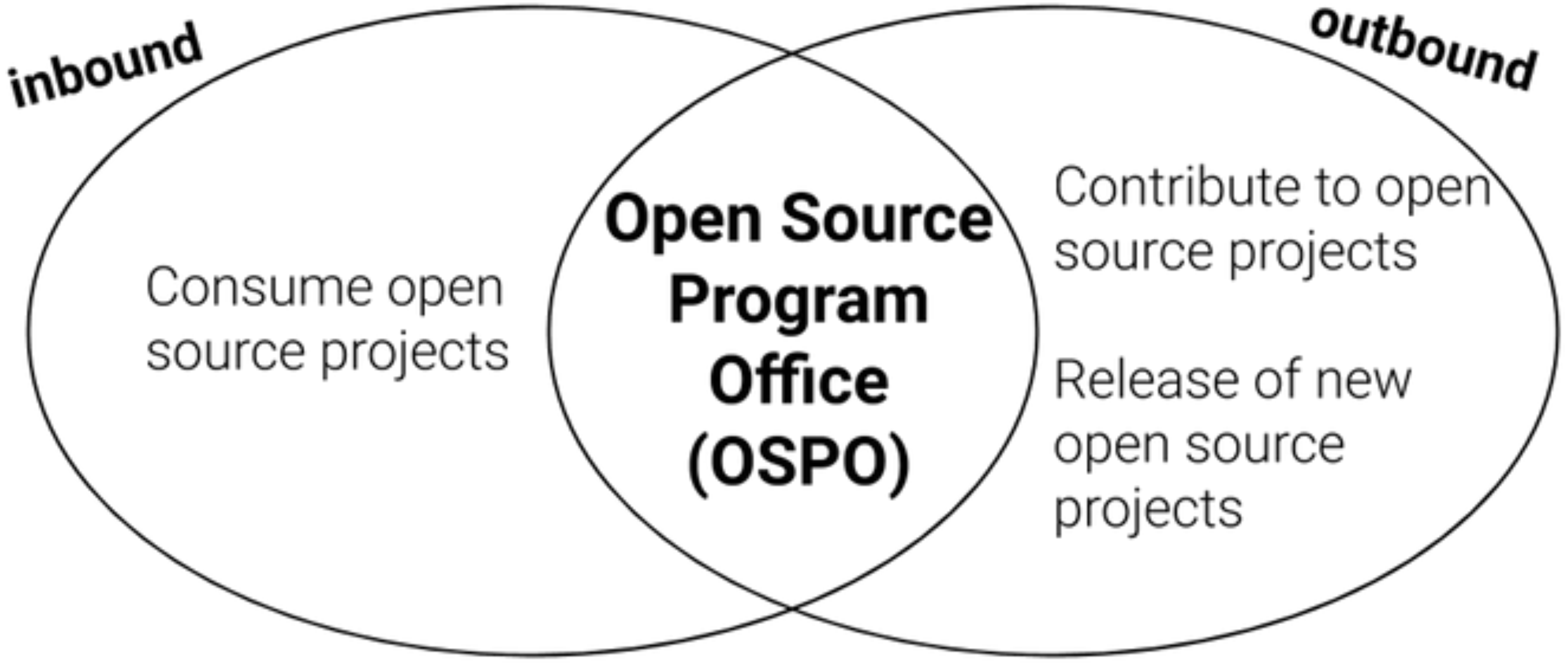
Open source program office

- Companies create Open Source Program Offices (OSPO) to manage their relationship with the open source ecosystems they depend on.
- By understanding the company's open source ecosystem, an OSPO is able to maximize the company's return on investment and reduce the risks of consuming, contributing to, and releasing open source software.
- Additionally, since the company depends on its open source ecosystem, ensuring its health and sustainability shall ensure the company's health, sustainable growth, and evolution.

[A guide to setting up your Open Source Program Office (OSPO) for success, J. Manrique Lopez de la Fuente, OpenSource, 2020]

Open source program office

- Some companies contribute to existing OSS projects.
  - That contribution could be part of the company's requirements for their solutions that need certain fixes in upstream projects.
  - For example, Samsung contributes to certain graphics-related projects to ensure its hardware has software support once it gets into the market.
  - In some other cases, contributing to OSS is a mechanism to retain talent by allowing the people to contribute to projects different from their daily work.
- Some companies release their own open source projects as an outbound OSS process.
  - For companies like Red Hat or GitLab, it would be expected. But, there are increasingly more non-software companies releasing a lot of OSS, like Lyft.

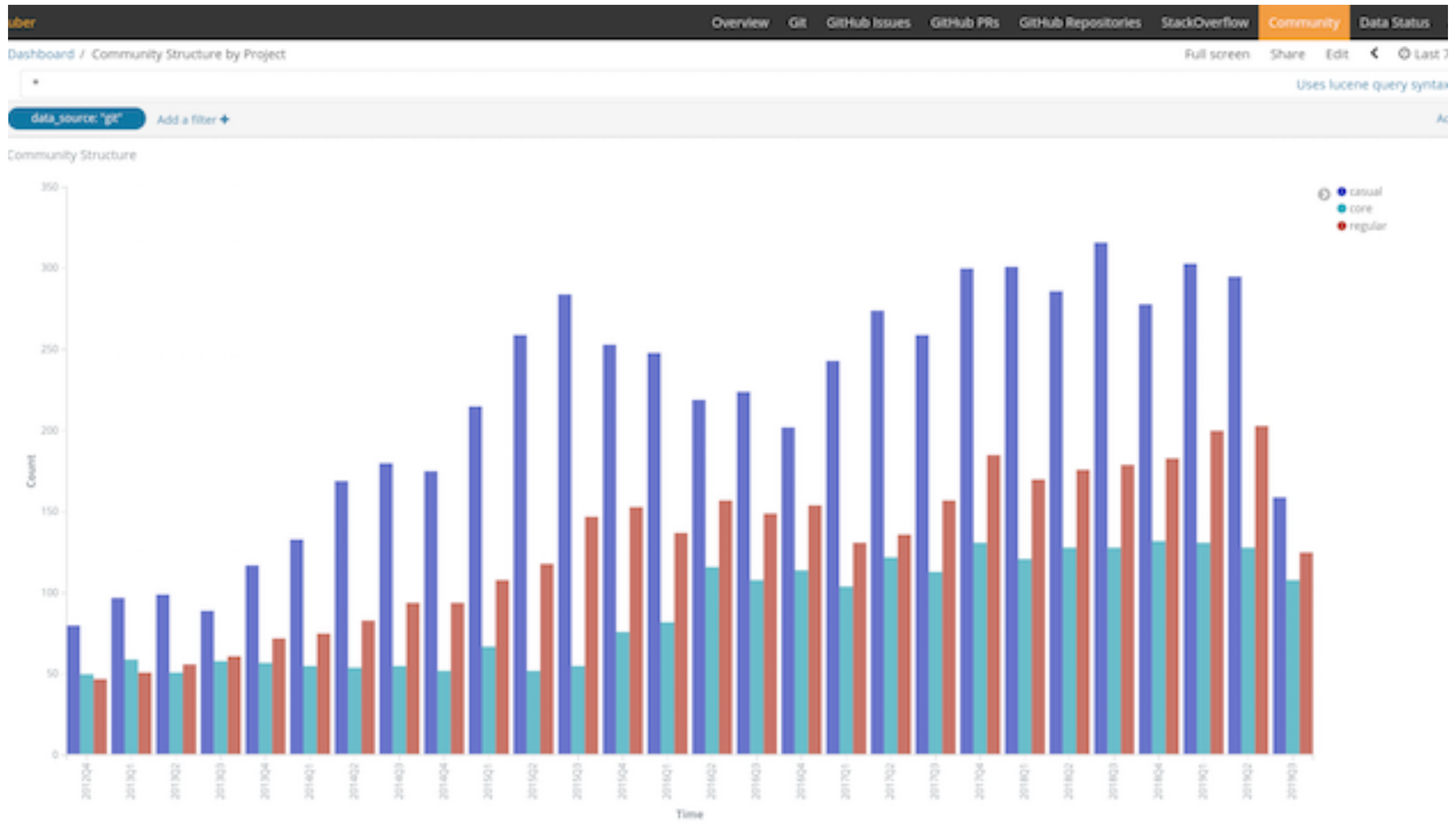

inbound
outbound
Consume open source projects
Open Source Program Office (OSPO)
Contribute to open source projects
Release of new open source projects

[A guide to setting up your Open Source Program Office (OSPO) for success, J. Manrique Lopez de la Fuente, OpenSource, 2020]

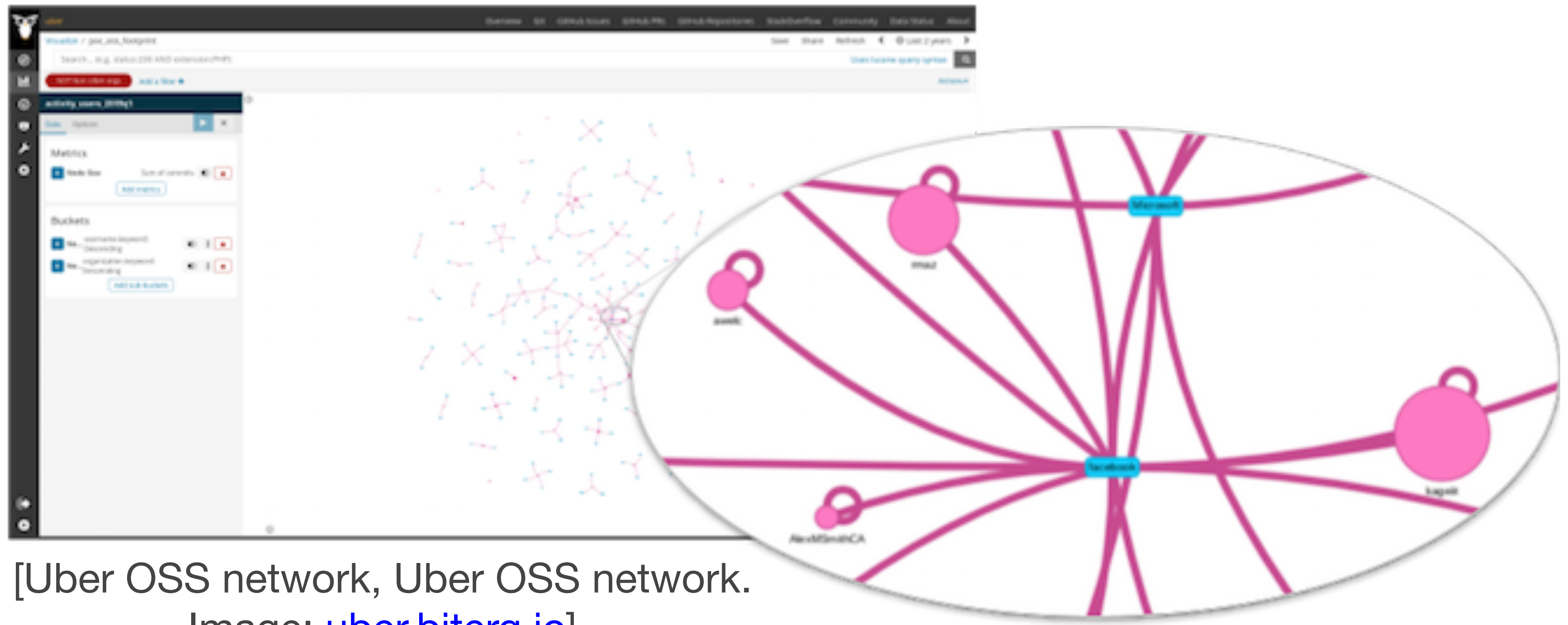
OSPO managers

S4Lab

- OSPO managers need to report a lot of information to the rest of the company to answer many questions related to their OSS inbound and outbound processes, i.e.:
  - Which projects are we using in our organization?
  - What's the health of those projects?
  - Who are the key people in those projects?
  - Which projects are we contributing to?
  - Which projects are we releasing?
  - How are we dealing with community contributions? Who are the key contributors?



[Uber OSS code core, regular, and casual contributors evolution. Image: uber.biterg.io]



[Uber OSS network, Uber OSS network. Image: uber.biterg.io]

[A guide to setting up your Open Source Program Office (OSPO) for success, J. Manrique Lopez de la Fuente, OpenSource, 2020]

OSPO community

- The TODO Group is "an open group of companies who want to collaborate on practices, tools, and other ways to run successful and effective open source projects and programs.”
  - For example, they have a complete set of guides with best practices for and from companies running OSPOs.
- The CHAOSS (Community Health Analytics for Open Source Software) community develops metrics, methodologies, and software for managing open source project health and sustainability.

[A guide to setting up your Open Source Program Office (OSPO) for success, J. Manrique Lopez de la Fuente, OpenSource, 2020]

The principle “community before code”

- If you have great code and a dysfunctional community, people will leave and the code will atrophy.
- If you have dysfunctional code but a great community, people will improve the code.
- That observation extends to the culture of your own company, where it becomes crucial to create a community among developers from different teams and let them work productively in the larger project community.
- You may need new structures to adopt this culture.

[Open Source in the Enterprise, Andy Oram and Zaheda Bhorat, O'reilly, 2018]

Why OSS Policies are a Must To Avoid Legal Risk (Tainting)

- Perhaps the biggest risk in using OSS is that it may impact proprietary software, including the potential requirement to make the source code for that software available to others.
- This is often referred to as OSS “tainting” of proprietary software.
- There is a growing trend in the enforcement of OSS license compliance. The trend is a movement from enforcement by OSS advocacy groups (such as the Free Software Foundation or the Software Freedom Law Center) to enforcement by commercial entities against other companies.

[Open Source Software Policies – Why You Need Them And What They Should Include, James G. Gatto, National Law Review, 2019]

Why OSS Policies are a Must To Avoid Legal Risk (OSS Concerns with SaaS)

- Under the GPL licenses, and many other OSS licenses, obligations that can result in tainting are triggered when software that contains or is derived from the GPL code is distributed.
- However, a growing number of OSS licenses (e.g., the Affero GPL license) include obligations that are triggered when such software is accessed by a third party over a network.
- For these "network access" licenses, obligations may be triggered by running OSS in a cloud or SaaS deployment, even if such OSS is not actually distributed.
- Due to the fact that with most cloud-based deployments the software is not distributed, many developers are lulled into a false sense of security that there are no OSS implications with such deployments.
- The reality is there are a growing number of OSS licenses that have significant legal implications, even when the OSS is not distributed, but accessed over a network.

[Open Source Software Policies – Why You Need Them And What They Should Include, James G. Gatto, National Law Review, 2019]

Why OSS Policies are a Must To Avoid Legal Risk (New Use Cases)

- Legal ramifications of using OSS under any particular license depends on the use case.
- Typically, running OSS internally within an organization, without distribution or third party access, imposes few if any legal obligations. Often, these uses are routinely approved by OSS policies.
- Future business plans may change this use. The OSS may later be packaged and distributed (e.g., white-labelled) or the OSS may be used to run an online service for third parties.
  - A change in use case may trigger different legal obligations depending on the terms of the relevant OSS license.
  - These future uses may cause problems if the OSS legal issues are not analyzed as this shift in business strategy occurs.
- If there is no policy in place to revisit the suitability of OSS as use cases change, unintended consequences can result.

[Open Source Software Policies – Why You Need Them And What They Should Include, James G. Gatto, National Law Review, 2019]

Why OSS Policies are a Must To Avoid Legal Risk (Patent Issues With Open Source Licenses)

- Significant patent issues can arise with OSS licenses.
- Many OSS licenses include express patent license grants and some arguably trigger an implied license.
- Certain OSS licenses require that you grant others a patent license relating to the use of certain OSS Components, any modifications you make and/or software in which the OSS components are included. In some cases, the license extends only to the OSS Component and/or modifications.
- In other cases, it can extend more broadly to software that includes the OSS component. Some patent license grants cover existing patents, but some also cover future acquired patents.
- Certain OSS licenses seek to deter a licensee from asserting certain patent infringement claims relating to the use of the OSS components by terminating the licensee's rights to use the OSS if it makes such an assertion.
- These provisions are often referred to as patent retaliation clauses.

[Open Source Software Policies – Why You Need Them And What They Should Include, James G. Gatto, National Law Review, 2019]

Security in open source ecosystem

- What do you think about open source code security?
- Is it more secure than the closed source one? why?

Security in open source ecosystem

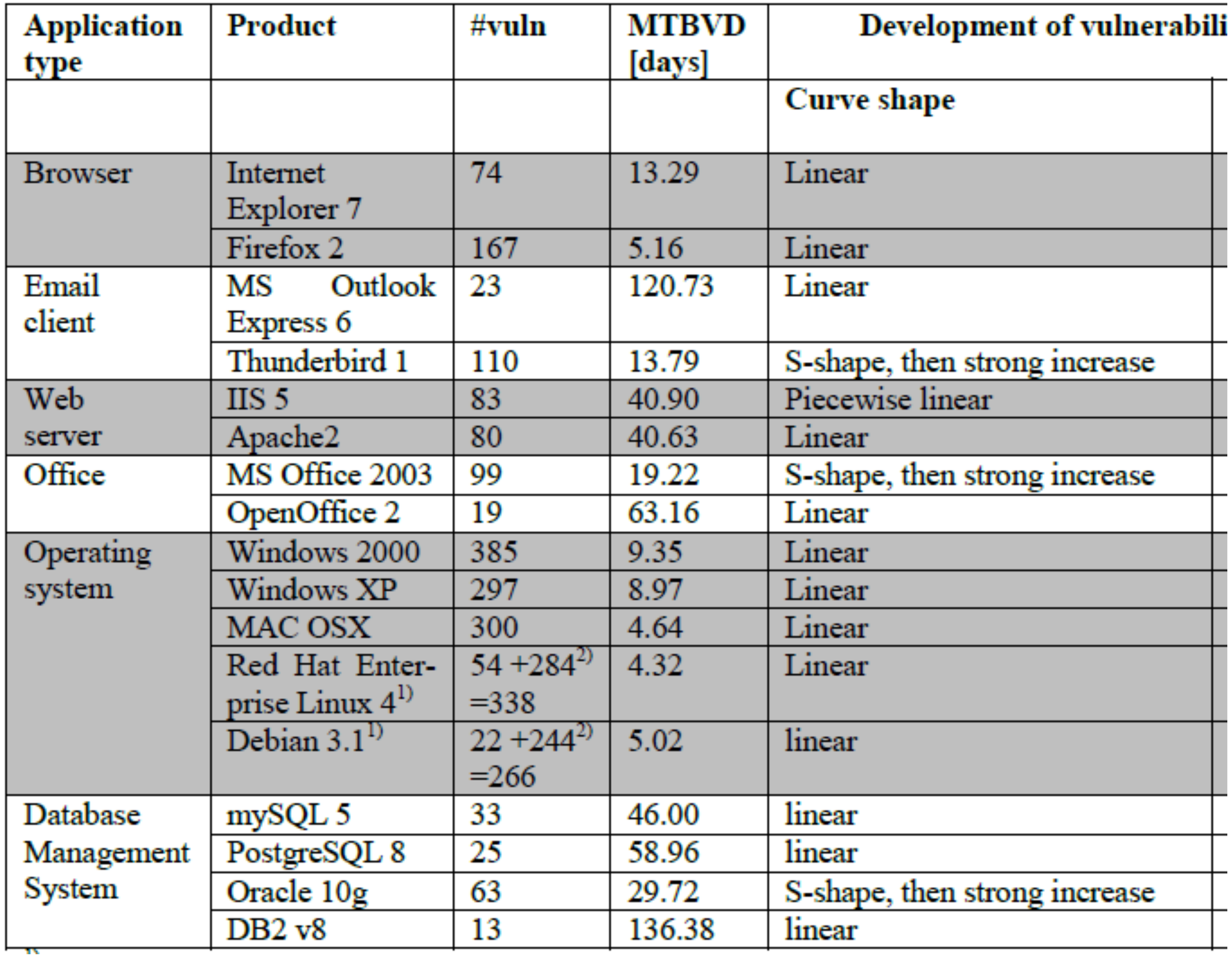
S4Lab

There are paradoxical facts about this!
Let’s see some.

Closed source is more secure because…	Open source is more secure because…
There is a specific team working on it	No one knows about the black box
No responsibility of open source codes	Peer-reviewed code model
Fast patch	Did they really patch it? Correct safe patch?

Overall observed vulnerabilities

- All codes are written by human developers! With same probability of vulnerability. (Not true anymore!)
- In an empirical study, no significant difference between the severity of vulnerabilities in open source and closed source software.
- So the problem is not which is better!
  - As many companies use both of them simultaneously.



Application type	Product	#vuln	MTBVD [days]	Development of vulnerabili
				Curve shape
Browser	Internet Explorer 7	74	13.29	Linear
	Firefox 2	167	5.16	Linear
Email client	MS Outlook Express 6	23	120.73	Linear
	Thunderbird 1	110	13.79	S-shape, then strong increase
Web server	IIS 5	83	40.90	Piecewise linear
	Apache2	80	40.63	Linear
Office	MS Office 2003	99	19.22	S-shape, then strong increase
	OpenOffice 2	19	63.16	Linear
Operating system	Windows 2000	385	9.35	Linear
	Windows XP	297	8.97	Linear
	MAC OSX	300	4.64	Linear
	Red Hat Enterprise Linux 4[1)]	54 +284[2)] =338	4.32	Linear
	Debian 3.1[1)]	22 +244[2)] =266	5.02	linear
Database Management System	mySQL 5	33	46.00	linear
	PostgreSQL 8	25	58.96	linear
	Oracle 10g	63	29.72	S-shape, then strong increase
	DB2 v8	13	136.38	linear

[Security of open source and closed source software: An empirical comparison of published vulnerabilities. Schryen, G., 2009]

How do you use it?

- If you as an enterprise are using open source, it has the tendency to be more secure, as the code is visible.
  - You have a pure code, with fewer/less documentation and quality assurance processes.
  - If you find a Vulnerability in the code, there would be no responsible man to patch it.
- You should have your own security program for such open codes.
  - You can use third party composition analysis and security scanners, but this only check the code against known vulnerability.
  - Bring the code into your SDLC security checks, including: code review, security tests, and etc.
  - Establish an open source policy with the right scope that uses an enforceability instrument.
  - Create a risk profile for open source software (OSS).
  - Perform performance tuning, and etc.

[6 ways to secure open source in enterprise, Gilles Gravier and Reza Alavi, Wipro, 2020]

[Is Open Source Software More Secure than Proprietary Products? Hilton Collins, GovTech, 2010]

Scalability of using open source codes

S4Lab

- In an enterprise environment, the sheer volume of machines makes any change in operating systems and applications a costly and time consuming undertaking
- Whether you’re switching to open source for servers, the desktop, applications, or all of the above, you should first test all of the new software thoroughly in a lab environment and then run a pilot program with one department or group of users before rolling out the change on a large scale
- Technical support may not be provided by the vendor, or may cost extra.
  - Also commercial distributions of open source products that do include tech support, but their cost is not zero and may even approach or exceed that of proprietary software
- Also administrative overhead may be also greater.

[https://www.techrepublic.com/article/how-scalable-is-open-source/]

QA