The Evolution of Open Source

Analyzing the values and benefits from our shared commons

By Mike Nolan

Introduction

Since the 1970's the production of software has evolved down a variety of paths. As digital technology development accelerated at an exponential rate, firms, governments, and individuals alike began investing labor into utilizing digital technology to solve a variety of problems. In this essay, I will explore the traditional oppositional structure posited by Eric Raymond between open source and proprietary production of software. Then, I will analyze how the software landscape has fundamentally changed, altering the relations of these two modes of software production and most importantly, how this has impacted the value generated from digital infrastructure which plays an increasingly vital part in our daily lives.

Situation

In the early days of computing, both software and hardware platforms were far from unified. Many competing firms would invest significant funds into developing similar platforms and products simply so that they could build their differentiating factor on top of them. The hardware was so expensive, the software would often be given for free such as fortran on early IBM machines (International Business Machines Corporation 2008).

In the 70s and 80s, computing hardware started to become standardized in order to streamline methods of hardware development. This happened to a point where pieces of software could be run across many different hardware platforms. When computing hardware systems became generalized, (x86) software's purpose became much more valuable across the market and became commoditized. At this point, walled gardens began being built with varying levels of interoperatability. Firms would develop software systems to lock in downstream developers into their systems. Whether it was DirectX vs OpenGL, posix vs DOS, users and developers alike were forced to choose which garden they would like to reside in.

It was during this period of software commoditization that Richard Stallman (and others) began positing an alternative form of software development and distribution. Born out of an issue with struggling with proprietary printer drivers (Free Software Foundation 2018), Stallman took issue with the lack of ownership associated with software which runs on various devices. From this, the Four Freedoms were written, leading to the GNU projects, Free Software Foundation, GPL, Linux, and much more.

As Free software grew in popularity, some large software firms took issue with this new method of production and distribution as it directly contradicted their proprietary methodology of cornering markets and locking in customers to their eco-systems. Microsoft was known for calling Free software "communist" and a "cancer on all intellectual property" (Amy Harmon and John Markoff 1998).

Through this contradiction, the conflict between Free software communities and proprietary enterprises grew. Particularly the viral nature of copyleft licenses used by Free software

projects scared many firms from allowing their use, resulting in company-wide bans across many copyleft projects. Many in the Free software community we're sympathetic to firms yet still wanted to share code as they believed it was beneficial to business and markets (as it cuts down duplicate costs and promotes competition). Through these sympathies, the more "business-friendly" term "open source" was coined and with it a series of more permissive licenses (Bruce Parens and Eric Raymond 2007). These permissive licenses, like the copyleft, released software "as is" without warranty. However, permissive licenses do not have any requirements of also sharing derivative works or linked software.

With these new "permissive" licenses, firms were able to adopt open source software in order to help cut costs in building out their core product offerings. Over time, firms even contributed back to projects and began their own as the benefits of cost-sharing became realized throughout the business community and as engineers increasingly depended upon this method of software production.

As software firms have grown and coalesced into just a handful of large international conglomerates, the purpose of a software firm has expanded into a variety of industries such as

- Knowledge/Insights
- Hardware licensing
- Internet/Connectivity

Many of these firms have achieved market dominance by acquiring large sections of the supply chain for the production and consumption of the software they build. A firm such as Google not only collects and indexes nearly the entire internet but also acquires various firms in automotive (self-driving cars), home management (nest), etc. **By owning such a large portion of the supply chain, products are able to take advantage of each other and build "ecosystems" that give a unique benefit to products owned by the governing firm.**

Many of these ecosystems take advantage of open source software and even deeply rely on open source communities' continued support of the software they are building. Firms are incredibly interested in the cost savings which come from an open source mode of production but struggle with the increased precarity generated by the limitation of liability or ownership of key parts of their supply (value?) chain.

Two Structures of Software Production

As Eric Raymond details in his book "The Cathedral and the Bazaar", proprietary software development in corporations was built in significantly different ways than that of open source software. These methods of production each produced distinct types of value which were captured differently by various stakeholders involved.

The Cathedral

The Cathedral is a centralized method of developing software that places the intellectual property as a closely guarded resource whose access is closely monitored. Firms sell access or use of their IP through licensing schemes which act as their primary source of revenue. Firms will also hire laborers to develop their IP and claim ownership to any output generated

by their laborers. **A firm's primary goal is to develop an IP valuable enough that customers will pay for access. The Cathedral is a methodology that most closely resembles traditional manufacturing. Firms will hire labor to design and produce goods which are then paid for and bought by consumers. Firms will dictate their actions based upon predictions of consumer demand.**

Firms that develop software applications (like traditional manufacturing) must either make or acquire a significant investment in order to develop economies of scale. While the distribution and replication of software have a near-zero cost (Weber 2005), the production of these complex systems can be quite expensive, requiring significant amounts of loss leading prior to making their investment back. This means that proprietary software products often have to target large markets in order to be profitable.

The nature of software, however, means that the IP, as well as the end result product which is consumed, are one and the same. To sell the good is to give direct access to the IP as well as the ability to re-produce the manufactured output of the firm with little to no marginal reproduction costs. Firms, therefore, utilize safeguards to discourage such action through strict IP law and lobbying firms, DRM software, and other technological safeguards which make reproduction difficult.

The Cathedral is primarily built by a single organization with a unified vision and goal. This creates a cohesive end product that oftentimes suits the needs of the most demanding consumers. However, the unified vision often comes at a cost of use and accessibility to secondary consumers who do not have as large of a market. The Cathedral is an old form of capitalistic production, well studied and researched by business and economists alike. It was the most intuitive method of producing software when it became commoditized. However, it struggles with the contradictions created by the nature of software's near zero reproduction cost.

The Bazaar

The Bazaar is the system of production which represents the early open source software communities of the period in which Raymond wrote his book. Communities were often times reasonably small, even for popular pieces of software. Instead of the centralized organization of the firm described above, communities were decentralized, each community focusing on the development of a singular tool with a constrained purpose rather than a unified ecosystem. The producers and consumers of value in The Bazaar are much closer in relation and often times the same. The most powerful entities within The Bazaar are those who invest the most amount of time in the creation of value.

Due to the decentralized nature of The Bazaar, it is largely upon the consumer to combine and configure various tools into a cohesive package in order to be of use. In comparison to The Cathedral, the decentralized and collective nature of The Bazaar means that individuals do not have the investment to build complex systems on their own and later recoup their costs. Instead, producers within The Bazaar will prioritize contributing small bits of

infrastructure, using the existing available infrastructure within the commons to do so. As more contribute to The Bazaar, the possibilities of new contributions will become slightly more complex and advanced. While the lack of up front investment often means more utilitarian qualities in open source software, the nature of infrastructure created means a more comprehensive base of utilities and configuration.

While The Bazaar is decentralized, it is not necessarily any more democratic than The Cathedral. **While it is upon each community to determine how it is governed, many tend to implicitly follow a feudal structure. A singular or small council of leaders will have final say over what changes are accepted and the road-map of the project. If the whims of the leader do not suit that of the community, the community may fracture or all together leave to start a separate project which better suits their needs**.

The objectives of The Bazaar are different from The Cathedral in that the growth of value is only encouraged to fit the individual community member needs. Rather than the imperative of infinite revenue growth aspired to by the firm, a singular community member will generally only contribute as much labor as they feel necessary to fulfill their own needs. Also, unlike The Cathedral, the co-operative nature of The Bazaar eliminates the need to protect IP. Instead, the valued resource is the labor of contributors. This resource is often in a precarious position due to outside forces such as their job, social life, etc. Communities have attempted to safeguard this resource through non-profit funding strategies (grants, foundations, small donations) and limiting the amount of community support given which can often hinder entry of less technical members.

These two modes of production produced directly competing products. **Linux or Windows, MS-SQL or Postgres, Internet Explorer or Firefox. These products competed against each other at all parts of the software stack, from operating system, to software frameworks/packages, to end user applications.** The existence of open source products directly competing with the ones produced by firms meant that ecosystems formed around certain sets of compatible products (for instance, the Microsoft ecosystem including Windows, Office, Defender, etc.) When a user “buys in” to the ecosystem, they are heavily encouraged to only use products offered within that system. This unilateral strategy employed by firms segmented the market and prevented cost (and benefit) sharing among various actors (Zauberman 2003).

This market segmentation meant that open source software was often relegated to exclusive use by people who had the situational and technical capability of producing software. **These people must either be highly technical, often with particular needs from their software tools not experienced by general users or people who could not afford to buy in to proprietary ecosystems.**

Open source software’s nature of free (re)distribution also implies an inherent break in the chain of liability which is common in proprietary software systems. The common structure of software vendor relations used by firms and states alike (via public/private partnerships) creates an inherent contradiction for open source software development. Therefore, open source software communities often lack large organizational buy-in and are sustained mostly by individuals.

Performance

The value produced by both systems was tools meant for the utility of their users. For proprietary products, the value was shared in the form of revenue for firms and utility for users. As firms developed exclusive ecosystems which rely on a suite of interlinked tools, the amount of value generated can be increased (and captured) through tools building off of each other.

The value produced by open source ecosystems, does not directly generate value in the form of revenue but rather only utility for users. While this makes it more difficult for open source production to exist in the presence of larger markets, the utility value produced is much greater because it does not need to be shared by two adversarial actors (firms and customers).

A New Situation Requires New Structures

With the advent of more interconnected services, larger networks of media and data distribution, and more of life becoming digitized (communication, travel, work) the nature of software has transformed. **While software tools were the primary commodity in the previous** structures, new commodities have become as indispensable as the tools. Specifically data, “insights”, and digital media. These new commodities have created markets much larger than those which are simply selling tools to consumers and with these new commodities, software firms have expanded their reach.

These new resources markets create an opportunity for the development of software to facilitate access. While this new software was not the primary commodity upon which the firm derived value from, it had utility. Specifically, the utility of facilitating access to various digital resources. The uniformity of this software had utility to a number of firms whose goal is to generate revenue through the monopolization of a digital resources. With this new situation, these firms have large incentives to share costs of platform development through contributing to open source ecosystems.

The Platform

The new alternative to classical software systems can be exemplified by the platforms that control much of our digital life today. These platforms are not necessarily commodities themselves but rather manage access to a commodity for users of various types. Facebook is a platform which manages access to communities and media for end users, and willing eyes, data and ad positioning for advertisers. Amazon is a platform which facilitates access to distribution networks and a global marketplace for producers and a network of goods and services for consumers. Google is a platform which facilitates access to information and digital services for end-users, and again, willing eyes and user networks for digital service providers and advertisers. The nature of the platform has positioned their creators to exploit multi-sided markets by acting as a mediator between various actors from various sides (Rochet and Tirole 2006).

Due to the nature of each of these platforms simply facilitating access to commodities, the exclusive access to the intellectual property of a large majority of their systems is not of

concern. Therefore, a large chunk of systems are released to the public free of charge and often collaborated on by a number of firms. These systems are generally built such that they can be configured to manage access to a variety of digital resources. Much of these systems are simply configurations of existing open source software tools. Their true value comes from allowing the deploying firm to effectively control access to their true valued commodity.

Just like with the creation of Cathedral like software ecosystems, firms must invest significant amounts of funds into not just building the digital platforms themselves, but also capturing and monopolizing the end result commodity which their platform facilitates access to. While the use of open source software has lowered costs of building the platform infrastructure, significant amounts of capital must be diverted to capturing users, workers, data, etc. in order to achieve a level of commercial viability. In the case of Uber, an investment must have been made into both the development of the app and corresponding services. However, a much larger investment was required in order to capture both a large enough number of drivers and customers to develop brand viability (Tina Bellon 2020). Only upon near-total monopolization of these end result commodities can an economy of scale effectively function and recoup its initial investment cost (David Wessels n.d.).

The Roads and Bridges

The nature of firms' adoption of open source software to facilitate access to their resources has greatly increased the amount of labor being directed towards the development of open source software. Where previously, individuals would develop tools that suit their personal needs and other individuals will contribute based upon their own needs, the nature of firm owned labor has been directed at enhancing only particular types of open source software (Yegulalp 2014).

The types of open source software firms would prioritize contribution to would be utilities which can help consolidate their own exclusive control of digital resources. This means that while most open source software developed previously would be driven by the needs and wants of individuals, now a significant portion of open source software systems developed is aimed at the needs of large organizations, particularly, firms.

Given that many firms capture value through managing access to digital resources, they generally seek to develop open source systems to facilitate that. These systems range in terms of functionality but often can be described as "digital infrastructure". This sort of infrastructure can be (but is not limited to):

- Server orchestration software (kubernetes, jenkins, etc.)
- Web servers (apache, nginx, etc.)
- Databases (mysql, mongodb, elasticsearch)
- Analytics & Machine Learning (Tensorflow, PyTorch)
- Backend frameworks (Django, Ruby on Rails)
- Frontend frameworks (React, Vue, Angular)
- Operating Systems (Linux)

While it is true that there have been development and use of competing platforms which are in themselves open source (such as Mastodon vs Twitter), these platforms often fail to compete due to being unable to access those same commodities monopolized by their

proprietary counterparts (e.g. Mastodon, f-droid, OpenStack).

The large but targeted investment of labor by firms in the production of open source software has created an unequal level of infrastructure with certain types of utilities being developed to a far greater capacity than others. As we can see from above, utilities that serve the development of large digital platforms are both advanced and well maintained. However, other utilities such as end-user software applications, particularly targeting non-technical users are uncommon and much more poorly maintained. This lopsided availability of infrastructure can create incentives to build upon infrastructure created for the purpose of digital platforms even when it is not necessarily ideal to do so.

Performance

The value generated by new proprietary software ecosystems (namely, platforms) is primarily usurped by firms in order to recoup investment costs while providing value to users through helping with distribution & access to that end commodity. While there is a genuine utility in making digital resources available through tools and platforms, the primary utility of the software ecosystems is really to facilitate access in order to allow value capture by the firm. The firm has total ownership of the underlying resource which is of primary use to the end-user and the end-user must pay in order to have any sort of access to that.

The value generated by much open source software is geared towards building large software systems meant to connect and **facilitate access** to digital resources. While this can be of use to individuals an a variety of ways, this value is most useful to large firms which already have access to a significant portion of digital resources (Srnicek and De Sutter 2017).

Firms deriving majority of their revenue from using digital platforms to mediate access to digital resources has incentivized cost sharing among maintaining the digital infrastructure upon which these platforms operate. This cost sharing has significantly cut down on the cost of running infrastructure, allowing firms to focus more time and money on capturing more and more digital resources and thus allowing their platforms to control access to additional resources which their users value.

The value generated through firms' decision to share infrastructure costs has not operated within a bubble. Firms' investment in open source infrastructure has contributed value to the larger open source software commons but that value does not benefit all equally. The value being generated by many new contributions to software commons ultimately benefit organizations which are able and seek to monopolize digital resources. Meanwhile, those looking to utilize the open source software commons to produce software systems that do not seek to control access to digital commodities will likely have to invest significantly higher amounts of capital in order to do so. As firms dictate more and more contributions to the open source "commons" the nature of what can be done within that commons will change.

Conclusion

Throughout the advent of the high speed internet within the developed world, the value of software as a commodity has been dwarfed by that of data and digital social networks. The business strategy of many large software firms has since shifted towards capture of these digital commodities. Due to these new incentives, firms have begun to share costs of the

building and maintenance of digital infrastructure that nearly all digital platforms rely on. While many claim the large injection of capital and labor into the building and maintenance of open source software means a more equitable means of software production (Finley 2016), the value has not been distributed equally among all actors.

In particular, software which addresses smaller markets not suitable for platforms (either due to ethical/legal issues or market size), or who serves a purpose which cannot be hammered into the model of managing access to another commodity, has had significantly smaller investment in the infrastructure needed to build and maintain these software systems. Open source software is often considered a "Do-ocracy". Those who "do" the most get to determine the governance and direction of the infrastructure created (CommunityWiki 2021). This method of production is not a democratic method. The value is determined by those who are able and willing to invest the largest amount of capital and labor. The introduction of well financed firms has created an imbalance in our digital infrastructure tilted towards the benefit of those controlling large software platforms and usurping labor value which might have otherwise been directed at other parts of the population. "Do-ocratic" governance of our open source software commons has allowed for not all voices to be equally important in determining the values generated by our labor.

Bibliography

Amy Harmon and John Markoff. 1998. "Internal Memo Shows Microsoft Executives' Concern Over Free Software." November 3, 1998. https://archive.nytimes.com/www.nytimes.com/library/tech/98/11/biztech/articles/03memo.html.

Bruce Parens and Eric Raymond. 2007. "The Open Source Definition (Annotated) | Open Source Initiative." March 22, 2007. https://opensource.org/osd-annotated.

CommunityWiki. 2021. "CommunityWiki: Do Ocracy." April 29, 2021. https://communitywiki.org/wiki/DoOcracy.

David Wessels. n.d. "What Will It Take for Uber to Become Profitable?" Knowledge@Wharton. Accessed August 7, 2021. https://knowledge.wharton.upenn.edu/article/uber-profitability/.

Finley, Klint. 2016. "Open Source Won. So, Now What?" *Wired*, August 11, 2016. https://www.wired.com/2016/08/open-source-won-now/.

Free Software Foundation. 2018. "'The Printer Story' Redux: A Testimonial about the Injustice of Proprietary Firmware — Free Software Foundation — Working Together for Free Software." April 26, 2018. https://www.fsf.org/blogs/community/201cthe-printer-story201d-redux-a-testimonial-about-the-injustice-of-proprietary-firmware.

International Business Machines Corporation. 2008. "IBM History." January 1, 2008. https://www.ibm.com/ibm/history/interactive/ibm_history.pdf.

Lerner, Josh, and Jean Tirole. 2005. "The Economics of Technology Sharing: Open Source and Beyond." *Journal of Economic Perspectives* 19 (2): 99–120. https://doi.org/10.1257/0895330054048678.

"Open Source Computer Programs: 'A New Communism' Claims Bill Gates! - L'Humanité in English." 2020. November 27, 2020. https://web.archive.org/web/20201127230254/http://www.humaniteinenglish.com/spip.php?

article319.
Raymond, Eric S. 1999. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. 1st ed. Beijing ; Cambridge, Mass: O'Reilly.
Rochet, Jean-Charles, and Jean Tirole. 2006. "Two-Sided Markets: A Progress Report." *The RAND Journal of Economics* 37 (3): 645–67.
Srnicek, Nick, and Laurent De Sutter. 2017. *Platform Capitalism*. Theory Redux. Cambridge, UK ; Malden, MA: Polity.
Tina Bellon. 2020. "Uber, Lyft Spend Big in California to Oppose Even Costlier Gig-Worker Law." *Reuters*, October 5, 2020, sec. APAC. https://www.reuters.com/article/uber-california-idUSKBN26Q2LX.
Weber, Steven. 2005. *The Success of Open Source*. Cambridge, Mass.: Harvard Univ. Press.
Yegulalp, Serdar. 2014. "Who Writes Linux? Corporations, More than Ever." InfoWorld. February 3, 2014. https://www.infoworld.com/article/2610207/who-writes-linux--corporations--more-than-ever.html.
Zauberman, Gal. 2003. "The Intertemporal Dynamics of Consumer Lock-In." *Journal of Consumer Research* 30 (3): 405–19. https://doi.org/10.1086/378617.