How Licences Learn

Madiha Zahrah Choksi*
James Grimmelmann†

Open-source licenses are infrastructure that collaborative communities inhabit. These licenses don't just define the legal terms under which members (and outsiders) can use and build on the contributions of others. They also reflect a community's consensus on the reciprocal obligations that define it as a community. A license is a statement of values, in legally executable form, adapted for daily use.

As such, a license must be *designed*, much as the software and hardware that open-source developers create. Sometimes an existing license is fit to purpose and can be adopted without extensive discussion. However, often the technical and social needs of a community do not precisely map onto existing licenses, or the community itself is divided about the norms a license should enforce. In these cases of breakdown, the community itself must debate and design its license, using the same social processes it uses to debate and design the other infrastructure it relies on, and the final goods it creates.

In this Article, we analyze four case studies of controversy over license design in open-source software and hardware ecosystems. We draw on Stewart Brand's *How Buildings Learn*, a study of how physical buildings change over time as they are adapted and repurposed to deal with new circumstances by successive generations of users. Similarly, we describe how open-source licenses are adapted and repurposed by different communities confronting challenges. Debates over license drafting and interpretation are a key mechanism of achieving the necessary consensus for successful collaboration. The

*. Ph.D. Student in Information Science, Cornell Tech. We presented earlier versions of this Article at the 2023 Fall Forum on Data in Business & Society at Lewis and Clark Law School, and the Digital Life Initiative Seminar at Cornell Tech. Our thanks to the organizers and participants, and to Aislinn Black, Eugene Bagdasaryan, Solon Barocas, Elettra Bietti, Ryan Calo, A. Feder Cooper, Charles Duan, Tabrez Ebrahim, Deborah Estrin, Alicia Gibb, Benjamin Mako Hill, Wendy Ju, Michael Madison, Ilan Mandel, Eben Moglen, Helen Nissenbaum, Madelyn R. Sanfilippo, Yan Shvartzshnaider, Salome Viljoen, Luis Villa, and David Gray Widder.

†. Tessler Family Professor of Digital and Information Law, Cornell Law School and Cornell Tech.

resulting licenses are the visible traces of the constant political work that sustains open-source collaboration. Successful licenses, like successful buildings, require ongoing maintenance, and the record of license changes over the years is a history of the communities that have inhabited them.

Introduction 2
I Open Licensing as Infrastructure 5
A How Commons Govern 5
1 Recursive Publics 6
2 Knowledge Commons 8
3 Value Articulation 10
B How Buildings Learn 12
C Crossing the Streams 14
1 From Architecture to Architecturalism 14
2 The High Road and the Low Road in Open-Source Licensing 16
II Case Studies 18
A Open Source Software 19
1 Drafting GPLv3 21
2 OpenOffice and LibreOffice Split 26
B Open Hardware 31
1 The Closing of Makerbot 34
2 The Arduino Trademark Tussle 37
III Designing the License, Designing the Community 43
A Lessons from Open Ecosystems 44
B Community Building and Governance 47
Conclusion 49

Introduction

Buildings are infrastructure that adapt to changing circumstances over time. Buildings do not endure simply because they are solidly built. Foundations shift; timbers rot; pipes burst. Constant decay requires constant maintenance. And no building will be preserved unless it meets the ongoing needs of the people who occupy it. As needs shift, buildings themselves evolve to meet them.[1] Stewart Brand describes these as buildings that *learn*. Buildings live

1. Stewart Brand, How buildings learn: What happens after they're built (Penguin 1995).

and thrive when they learn about their inhabitants, their communities, and their ecosystems. A building that does not learn eventually becomes a lifeless skeleton.

Digital infrastructure, too, evolves over time. While it is maintained in different ways, and hence learns in different ways, digital infrastructure also has the capacity to adjust to meet to its communities' changing needs. As stakeholders of open technical ecosystems, members of technical communities are commonly guided by a commitment to producing and sharing technologies they co-create. Some technical infrastructure is designed strategically and built to last.[2] It is carefully stewarded to ensure rock-solid stability and backwards compatibility. Other infrastructure is designed without much foresight, and serves short-term goals,[3] but its widespread adoption means that it is subject to constant and responsive evolution. In both cases, technical innovation and social governance is driven by diverse and evolving needs and demands.

We claim that legal infrastructure also evolves as it learns about its users. Although our focus is on digital communities that produce open-source software and hardware, we draw inspiration from the physical adaptation of buildings. Ideas and innovation are the lifeblood of open-source communities, and intellectual property (IP) rights play a crucial role in defining these communities' identity and safeguarding what matters most to them. An IP license is community infrastructure; it structures collaboration (and competition) between members. But rather than permanently freezing in place permissions and restrictions, the licensing process involves continuous adaptation to fit the dynamic landscape of innovation. The license that governs a community's IP rights not only establishes rules of conduct but also becomes a symbolic representation of the community's essence. Just as carpenters and masons use saws and cement mixers to reconstruct and retrofit buildings, open-source developers use IP rights and IP licenses to understand, adapt, and govern their innovations.

Open licenses protect creative innovations and formalize rules of engagement and use for a broad community of users. The designing community collaborates to write rules that describe how others can use, change, and share what they have produced. As a legal instrument, open licenses enable creators to grant specific permissions. As a social instrument, licenses represent

2. For example, the Hypertext Transfer Protocol was first published in 1989. HTTP has been widely adopted across the web, but the standards that comprise HTTP have developed and expanded over time under the stewardship of the IEFT HTTP Working Group.

3. For example, hashtags are adopted in uncoordinated ways and were created ad-hoc by users of Twitter.

much more: the summation of community values and ethics through a long process of discussion and deliberation. Creative Commons (CC) licenses, for example, were developed in response to the traditional and restrictive nature of traditional copyright law.[4] The overarching goal of the CC license suite is to liberate work products, to foster creativity by building on top of and remixing other work in a variety of formats.[5]

In a parallel illustration of this phenomenon, the online gaming community offers valuable insights. In 2023, the online gaming community unequivocally rejected the latest iteration of the Open Gaming License (OGL). The revised license prioritized brand development over embracing community norms and fostering growth, compromising its effectiveness.[6] More importantly, alterations in the revised license's language failed to address the goals and motivations of the gaming community, which perceives online gaming as an environment for establishing connections with like-minded individuals and friends.[7] For the open gaming community, the licensing process itself serves as a platform for defining and upholding core values.

Drafting a license is often a collaborative endeavor. Communities discuss, in stages, how to share their creative output with others. The process is slow and iterative, filled with moments of convergence and consensus, as well as moments of strong disagreement and debate.[8] Throughout this process, the community negotiates the parameters of sharing and use and instills the shared values and goals of their designs within the license. In this deliberative approach towards agreement, each debate and controversy is part of a *functional design process* that signals the community's values and motivations. Collaborative articulation becomes the arena through which the community enacts its shared values.

Designing a project involves a process of formulating the collaborative undertakings of contributors. This may encompass the creation of software applications, artistic works, or other collective endeavors, involving intricate

4. Lawrence Lessig, *The Creative Commons Commentary*, 65 Mont. L. Rev. 1—14 (2004).

5. *Id.*

6. Jess Weatherbed, *Dungeons & Dragons Finally Addresses Its New Open Gaming License - The Verge* (2023), https://www.theverge.com/2023/1/13/23554014/dungeons-and-dragons-dnd-open-gaming-license-announcement-wotc-hasbro.

7. D&D is a highly creative and collaborative video game. Players form relationships with characters, imagine unique objectives, and create new roles and scenarios. The community developed around the game, and the engagement motivates further imagination and creativity. From fan blogs and magazines the community became recursive, feeding new ideas and objectives back into the infrastructure of the game and how it could be imagined and played.

8. E. Gabriella Coleman, *Three Ethical Moments in Debian* (2005), https://papers.ssrn.com/abstract=805287.

planning and conceptualization. Necessary activities include brainstorming, defining objectives, and formulating the overall structure and features of the project. Drafting a license for a project, then, creates the legal terms and conditions that govern the use and distribution of the project's intellectual property. This could be a proprietary license, terms of service, open-source license, or other legal instrument that details how others can use and build upon the project. There is an isomorphism between designing the project and drafting the license, where the community expends considerable efforts in both endeavors. The license becomes an explicitly designed form of infrastructure for the community. Licenses, then, serve two goals: (1) to perform the function of a license in legal contexts, and (2) to codify community agreement and values. This Article examines the goals and values within communities that design. It provides a framework for understanding communities by tracing how they self regulate their intellectual property.

Part I of this Article describes the existing theoretical frameworks on which we draw. Part II describes four case studies of open hardware and software projects and communities: GPLv3, OpenLibreOffice, Makerbot, and Arduino. And Part III brings the theory to bear on the case studies, showing how they illustrate a process of infrastructural governance that is simultaneously collaborative, contentious, and political.

I. Open Licensing as Infrastructure

We are far from the first scholars to think about the infrastructure of open source. A large body of literature studies how communities form around open-source projects and govern the informational resources that they hold in common. Before we can make our argument that licenses too are a form of communally governed common infrastructure, we first survey some key lessons of the existing literature on open-source communities (Section A). We then turn to Stewart Brand's teachings in *How Buildings Learn* to describe how physical buildings are continually adapted to new community needs (Section B). Finally, we bring these two strands of work together to show how licenses emulate the same processes as buildings (Section C).

A. How Commons Govern

The starting point for any discussion of commons governance – physical or informational – is Elinor Ostrom.[9] Her work challenged the idea that shared

9. Elinor Ostrom year, Governing the Commons: The Evolution of Institutions for Collective Action.

resources, or commons, are necessarily overused and depleted. Through empirical investigations, Ostrom demonstrates how communities develop systems for managing common-pool resources. There are a number of design principles that contribute to successful commons management: clearly defined boundaries for resources, rules and regulations tailored to local conditions, monitoring and sanctioning mechanisms to ensure compliance, as well as mechanisms for conflict resolution and collective decision making.[10] Effective commons management requires participation from local communities in both design and implementation processes.[11] Polycentric governance is also crucial.[12] In this model, multiple levels of authority and decision making are coordinated to manage the shared resource. Polycentric governance systems are flexible, and can adapt to changing social demands.[13]

1. Recursive Publics

Ostrom emphasizes the shared governance institutions that are required for a sustainable commons. Later scholars have shown how the governance institutions of open-source communities have an interesting and important form. In Chris Kelty's formulation, they are "recursive publics." A public is a collective whose members address each other at large,[14] literally through publication – making public – and which understands itself as collective, distinct from other sources of power.[15] A recursive public is a public that is "vitally concerned with the material and practical maintenance and modification of the technical, legal, practical, and conceptual means of its own existence as a public."[16] It manages the infrastructure on which it depends (making it materially recursive), and it also understands itself in terms of its commitment to this maintenance (making it socially recursive). They are defined by their shared objectives, and their shared responsibility.

For a technical community, this recursivity is characterized by the ability to create and sustain the digital infrastructure – e.g., software, networks, and discussion forums – through which the community protects and promotes

10. *See generally id.*

11. Elinor Ostrom, *Beyond Markets and States: Polycentric Governance of Complex Economic Systems*, 100 Am. Econ. Rev. 641 (2010).

12. *Id.*

13. *Id.*

14. Benedict Anderson, Imagined Communities: Reflections on the Origin and Spread of Nationalism (rev. ed. 2006).

15. Charles Taylor, Modern Social Imaginaries (2003).

16. Christopher M. Kelty, Two bits: The cultural significance of free software 3 (2008).

openness. As they do so, they co-create new meanings and relationships.[17] They come to understand that particular infrastructural designs *are for* particular kinds of openness. For example, although many copyleft open-source licenses require only that source code be made available on request,[18] many projects using those licenses have a central canonical source repository on the Internet that includes not just the current source code but its complete modification history and extensive discussion. They understand openness to embrace a form of transparency that involves "working in public."[19] The community reasons about the infrastructure's purpose, works to challenge its limits, and finds new ways to express its goals in the infrastructure.

Another example of this recursive process is still visible within the Linux community. To address the hosting and distribution of Linux distribution software, volunteers host "mirrors" (i.e., copies) of the software for download. For example, when a decommissioned Linux mirror needed restoration in California, the Linux community rallied its resources on Twitter.[20] A lively discussion gained community interest, and through PayPal, contributors donated a few hundred dollars to help the volunteers purchase hard drives. In return, the volunteers promised to label each hard drive with the name of the "hard drive sponsor." The "hard drive sponsor" contribution tier sold out, and volunteers raised nearly enough to cover all hardware related costs.[21] However, what were the contributors actually supporting? The sponsored hard drive would sit "in a server, inside a locked rack, inside of a data center" where no one would actually see or acknowledge the sponsor's name.[22] Community members were driven by their commitment to what the tangible infrastructure of the Linux project represents. There is a mutually recursive feedback loop between the software infrastructure and the social infrastructure, where one process enacts the other and the cycle repeats to maintain the project.

Recursive publics not only create technical infrastructure but also build and maintain their own social infrastructure, as seen in the case of the Linux community on Twitter. The success of the Linux community relies not only on the technical quality of the software, but also on the social infrastructure

17. *Id.*

18. *The GNU General Public License v3.0* § 6 (2007), https://www.gnu.org/licenses/gpl-3.0.html.

19. Nadia Asparouhova, Working in Public: The Making and Maintenance of Open Source Software (2020).

20. Kenneth Finnegen, *Building the Micro Mirror Free Software CDN*, Life Kenneth (2023), https://blog.thelifeofkenneth.com/2023/05/building-micro-mirror-free-software-cdn.html.

21. *Id.*

22. *Id.*

that organizes around it. The community establishes norms for communication that include medium (IRC vs. forums vs. other modes), formats (what information should be included in a bug report), civility (what kinds of rhetoric are considered improper), and governance (who sets up the communications infrastructure and how they make decisions about it).

2. Knowledge Commons

Another important aspect of open-source communities is that they manage *information* common resources. They are dedicated not to the production of tangible goods like crops or oil, as in Ostrom's original examples,[23] but information goods that are non-rival and non-excludable. This difference creates both opportunities (because those goods can be freely shared without depletion) and challenges (because it can be harder to define and enforce communities boundaries and rules).[24]

Put another way, open-source communities create intellectual infrastructure.[25] This infrastructure supports the community's operations. It is also infrastructure for society at large – producing this infrastructure is often the point of the community in the first place. It includes knowledge (in the form of research, ideas, general purpose technologies, and languages); it enables cultural functions such as innovation, community co-creation, participation, and socialization.[26] These blended infrastructures draw upon both intellectual and cultural resources and showcase a symbiotic relationship between technical innovations and sociocultural contexts. For example, the open 3D printing community produces not only 3D print designs for hobbyists, but parts for printers themselves, documentation, and support guides.[27]

IP laws affect and structure these knowledge commons in important ways.[28] Most fundamentally, they create an underlying legal framework that determines which intellectual resources are subject to legal restrictions at all and which remain in the public domain.[29] Even where specific programs and de-

23. Year, *supra* note 9.

24. *See* James Grimmelmann, *The Internet is a Semicommons*, 78 Fordham L. Rev. 2799 (2009) [hereinafter Grimmelmann, *Semicommons*].

25. Brett M. Frischmann, Infrastructure: The Social Value of Shared Resources (2012).

26. *Id.*

27. Aaron Saenz, *Makerbot Is Asking You to Help Make More Makerbots* (2009), https://singularityhub.com/2009/08/17/makerbot-is-asking-you-to-help-make-more-makerbots/.

28. Brett M. Frischmann, Michael J. Madison & Katherine Jo Strandburg, Governing Knowledge Commons (2014).

29. Frischmann, *supra* note 25.

signs are subject to copyright or patent protection, for example, the knowledge that discussion about them embodies is generally not. Open software communities extensively utilize popular platforms like GitHub,[30] and Q&A sites like Stack Overflow.[31] Although both official technical forums (such as those used by Arduino, GPL, RepRap, or Linux) and hobbyist resources (such as those on YouTube, Reddit, TikTok, and Facebook Groups) outline their own rules and guidelines for engagement, they do not and cannot control how others may reuse, share, or build on the ideas exchanged there. This knowledge is inherently in the commons side of the information semicommons.[32]

Creative communities also frequently use IP licenses to create a "standing on the shoulders of giants" effect, explicitly allowing the practice of building new projects, software, or innovations by leveraging existing open-source resources, libraries, or codebases. This form of knowledge management and intellectual progress is unique to each community, and this knowledge creates downstream spillover effects that reciprocally benefit society. Knowledge sharing, therefore, is dependent on the relationships that form within their collaborative and culturally specific ecosystem. As observed in the Debian community, a nuanced understanding of how intellectual property laws can be adapted to self-govern resources within open communities is essential.[33] In other words, as intellectual infrastructures cultivate knowledge rich and productive social activities - such as peer production, they require adaptable tools that enable self-governance.[34]

30. Tsay, Jason & Dabbish, Laura & Herbsleb, James, *Let's talk about it: evaluating contributions through discussion in GitHub*, 2014 Proc. 22nd ACM SIGSOFT international symposium on foundations software engineering 144; Dabbish, Laura & Stuart, Colleen & Tsay, Jason & Herbsleb, Jim, *Social coding in GitHub: transparency and collaboration in an open software repository*, 2012 Proc. ACM 2012 conference on computer supported cooperative work 1277; Li, Renee & Pandurangan, Pavitthra & Frluckaj, Hana & Dabbish, Laura, *Code of conduct conversations in open source software projects on github*, 5 Proc. ACM on Hum.-computer Interaction 1 (2021).

31. Vasilescu, Bogdan & Filkov, Vladimir & Serebrenik, Alexander, *Stackoverflow and github: Associations between software development and crowdsourced knowledge*, 2013 2013 Int'l Conf. on Soc. Comput. 188.

32. Robert A. Heverly, *The Information Semicommons*, 2003 Berkeley Tech. L.J. 1127.

33. Coleman, *supra* note 8.

34. Yochai Benkler & Helen Nissenbaum, SSRN Scholarly Paper, *Commons-Based Peer Production and Virtue*, 14 J. Pol. Phil. 394 (2006).

3. Value Articulation

The growth and sustainability of open source projects are often assessed based on their ability to learn from community members, and vice versa,[35] leading to a healthy and thriving ecosystem.[36] E. Gabriella Coleman has shown how online communities define, understand, and govern themselves through continual rearticulation of their shared values. The process starts even before members join. When individuals choose projects to contribute to, there is a matching process between potential contributors and projects.[37] Project maintainers are driven by intrinsic goals related to the project and its overall ecosystem.[38] As users transition to becoming contributors, they undergo a socialization process, gradually adopting the identity of software craftsmen, which is marked by specific rites of passage.[39]

From the inception of a project, however, technical communities react to specific moments that alter or change the project's direction, motivation and commitments. Such ethical and political moments spark debate and controversy, but more importantly, these moments shift how the community can reimagine and transform their values. The Debian project[40] provides historical context about how community engagement and participation create ethical standards that signal and reflect community values. In the long and documented history of the Debian project [41], three key ethical and political moments emerge: (1) enculturation, (2) legal pedagogy, and (3) crises which help us understand how the community creates tools for managing their infrastructure. As these moments emerge, communal deliberation and

35. Sulayman K. Sowe, Ioannis Stamelos & Lefteris Angelis, *Understanding knowledge sharing activities in free/open source software projects: An empirical study*, 81 J. Sys. & Software 431 (2008).

36. Johan Linåker, Efi Papatheocharous & Thomas Olsson, *How to characterize the health of an Open Source Software project? A snowball literature review of an emerging practice*, 2022 Proc. 18th Int'l Symposium on Open Collaboration 1.

37. Qiu, Huilian Sophie & Li, Yucen Lily & Padala, Susmita & Sarma, Anita & Vasilescu, Bogdan, *The signals that potential contributors look for when choosing open-source projects*, 3 Proc. ACM on Hum.-Comput. Interaction 1 (2019); Karim R. Lakhani & Eric Von Hippel, How open source software works:"free" user-to-user assistance (2004).

38. R. Stuart Geiger, Dorothy Howard & Lilly Irani, *The labor of maintaining and scaling free and open-source software projects*, 5 Proc. ACM on human-computer interaction 1 (2021).

39. Nicolas Ducheneaut, *Socialization in an open source software community: A socio-technical analysis*, 14 Comput. Supported Coop. Work 323 (2005).

40. Coleman, *supra* note 8.

41. Debian is a free and open source operating system. It is the most popular distribution of Linux.

decision making processes enable the continued success and sustainability of the Debian project.

The first moment, enculturation, is about creating norms around shared resources and co-creation. To cultivate conflict-free deliberation and participation, the Debian community enacted social structures that guided the community towards open and effective communication. The community further developed and adopted a set of best practices suited to the project and established norms. More specifically, the publication of the Debian Social Contract enshrined a set of guiding principles for the Debian community. These included the community's commitment to creating and distributing free software, its expressive rejection of non-free software, and its commitment to open source ethos as a public benefit for the community.[42]

The second moment is legal pedagogy, where volunteer developers collaboratively contend with legal theory and pedagogy to define their understanding of freedom, particularly in terms of intellectual property law and free and open-source licensing. Legal theory influences developer values, and free-software communities actively participate in legal debates and discourse. In the context of the Debian project, participation requires at least a normative understanding of intellectual property law and free and open source licensing.[43] In 2006, Debian developers removed non-free portions of the project from the Debian archive due to concerns that distributing non-free software contradicted the project's commitment to free-software principles. This response was guided by concerns that the project's commitment to principles of free-software were contradicted by distributing non-free software.[44] Since the ethical implications of distributing non-free software posed normative and functional risks to the community's commitment to promoting free software ideals, this concern prompted a technical modification via community consensus.

The third moment is defined as instances of punctuated crises. Crises can arise around a number of issues such as project visibility, communication, size, and licensing. Within large open projects that rapidly scale, moments of "punctuated crises" are resolved through community engagement and deliberation.[45] In 2008, the Debian community raised concerns about the ethical implications of using a proprietary cloud computing platform, AWS. Through intentional participatory debates and discussions, the Debian community resolved how they see their project and the role of free software in an era of cloud computing.

42. Coleman, *supra* note 8.
43. *Id.*
44. *Id.*
45. *Id.*

The open and collaborative ethos of open ecosystems motivate developers and designers, and a mutually beneficial feedback loop further is nurtured by shared values and goals: reuse, iterative and collaborative development, and knowledge production and dissemination. These values are then enacted by the licensing frameworks that are written to meet the needs of diffuse communities. Each moment within the collaborative technical communities involved in open software and open hardware ecosystems represents a pivotal juncture where values, norms, and commitments undergo reimagining and transformation. The enculturation moment is characterized by the establishment of social structures, fostering an environment conducive to co-creation and shared resources. The legal pedagogy moment involves the community's engagement with legal theory and discourse, shaping their understanding of freedom and influencing their stance on intellectual property and licensing. Punctuated crises serve as catalysts for communal engagement and resolution, requiring active participation and deliberation to address issues such as project visibility, communication, and ethical implications. These moments collectively contribute to the resilience and sustainability of these ecosystems, reinforcing their dedication to open principles, collaborative development, and knowledge dissemination.

B. How Buildings Learn

Stewart Brand's Stewart Brand provides a complementary way of thinking about the management of infrastructure. According to Brand, buildings are evolving infrastructures. They are not static and unchanging, but have a life cycle with stages of development, adaptation, and renewal.[46] The relationship between buildings and their users is symbiotic, and a confluence of changing needs nudge buildings to grow and transform over time.

A building is made up of layers, which serve different functions, are easier or harder to modify, and require maintenance on different time scales.[47] A building's structure, for example – its foundation, pillars, and other structural elements – literally supports everything else in the building. It is difficult to change once built, and as a result, is often intended to require only preventative maintenance during the building's expected lifetime. Its services, on the other hand – such as its plumbing, electrical wiring, and heating systems – are more accessible and can be upgraded. Ripping and replacing knob-and-tube wiring is not cheap or easy, but a century-old house can be brought up to a modern electrical code, substantially changing how its rooms can be used. Services can and do fail; a homeowner can expect to need to replace a boiler

46. Brand, *supra* note 1.
47. *Id.*

every few decades. Even closer to the surface, furniture can be swapped out almost at will; a room can switch from a bedroom to a living room simply by changing a bed for a couch.

The layered nature of buildings allows them to be treated as a composition of interconnected and interdependent elements. Each layer serves a specific function and contributes to the overall performance and aesthetics of the building. The modularity of layers also enables architects and designers to approach building design in a more flexible style. Instead of designing a monolithic structure, they can focus on individual layers and how they interact with one another.[48]

Brand distinguishes two distinct modes in which builds are maintained and adapted through time: the "high-road," and the "low road."[49] Each mode represents divergent paths that architects and designers undertake when conceptualizing and executing architectural building projects.

High-road buildings are characterized by their permanence and ability to refine over time. They underscore sustainability, resilience, and aesthetic excellence. These buildings can be environmentally conscious, and resources are employed with purpose and strategy. Visually, these buildings tend to integrate with the surrounding environment. High-road buildings mature with time, and together, their age and their complex qualities contribute to "rich specialization."[50] Notably, high-road buildings are often culturally important and they are designed to signal that importance. They are therefore incredibly expensive.

An orthodox church, for example, was built, burned down, and rebuilt two more times. It becomes a cathedral, then a mosque, then a museum and then a mosque again.[51] Artwork, signage, services and skins are reimagined at each turn. The desire to maintain, support, extend and preserve the Hagia Sophia is constantly rearticulated by the desire to maintain its cultural and social value.[52] The building is a gathering place for many communities who recreate the same acts according to their own values.

Low-road buildings are less elegant. These buildings are designed improvisationally and with minimal stylistic consideration. They exist to be reimagined, thus producing high turnovers of both occupants and designs.

48. This interlocking layered modularity is an almost perfect fit for digital infrastructure. *See* Grimmelmann, *Semicommons*, *supra* note 24; James Grimmelmann & A. Jason Windawi, *Blockchains as Infrastructure and Semicommons*, 64 Wm. & Mary L. Rev. 1097 (2022).

49. Brand, *supra* note 1.

50. *Id.*

51. Robert S. Nelson, Hagia Sophia, 1850-1950: Holy Wisdom Modern Monument (Univ. of Chi. Press 2004).

52. *Id.*

Economic activity typically follows low road buildings, and these buildings often have lower financial barriers to access, such as rent. There is minimal care or concern over what actually happens in low road buildings, and they can be easily adapted and replicated. They are absent the preciousness that often is applied to high road buildings. On the low road, architects opt for standardized designs, materials, and methods that streamline the building processes. Nevertheless, low road buildings are uniquely powerful and inspiring. Their versatility is empowering, and they can quickly conform to changes in the natural environment or to serve an entirely new function. For example, garages can become office spaces or shops.[53] Or, temporary structures built in haste can transform into prominent incubating spaces and scientific research labs.[54]

C. Crossing the Streams

We believe that these two traditions bring complementary perspectives to bear. The work of maintenance unites them, and there are important connections in how communities carry out this work. As we will show, license choice and maintenance is an arena in which both types of work are visible. Indeed, almost from its beginning, Internet-law scholarship has appreciated the essentially architectural role of open-source licenses.

1. From Architecture to Architecturalism

An interesting strand in Internet-law scholarship connects the study of physical infrastructure to the study of digital commons management. Lawrence Lessig's famous slogan that "code is law" stands for the proposition that software can do the same regulatory work as law.[55] But the middle term in his syllogism was *architecture*: architecture is a regulatory alternative to law, software is architecture, therefore software is a regulatory alternative to law.[56] Lessig's argument drew heavily on architectural theory to make the crucial connection between software's and architecture's regulatory potential,[57] and

53. Brand, *supra* note 1.

54. *Id.*

55. Lawrence Lessig, Code: And Other Laws of Cyberspace (1999); *see also* Lawrence Lessig, *The New Chicago School*, 27 J. Legal Stud. 661 (1998); *id.*

56. *See* James Grimmelmann, *Note: Regulation by Software*, 114 Yale L.J. 1719 (2005) (discussing role of architecture in Lessig's argument).

57. *E.g.,* William J. Mitchell, City of Bits: Space, Place, and the Infobahn (1996).

a small but provocative follow-on line of legal scholarship took the architectural metaphor seriously.[58]

In particular, Lessig argued that the Internet's software[59] functions as a *constitution*; it makes crucial governance choices about who rules and embeds fundamental rights like the freedom of speech and privacy. In the Appendix to Lawrence Lessig, he explicitly linked this constitutionalism to architectural theory, discussing Michael Sorkin's Michael Sorkin.[60] Sorkin's quirky and elegant book is an argument about healthy and sustainable urban development styled as a zoning code. It prescribes the sizing and spacing of greenways, access roads, and other development to create a livable city. Lessig, in turn, argued that open-source software can embed the constitutional value of transparency in Internet architecture.

It is a modern truism that an open-source license functions as the "constitution" of its community.[61] But we think there is another metaphor that is equally apt: an open-source license is a *zoning code*. It describes what kinds of development are allowed: who can build what, where, and how. It does not and cannot require anyone to use the software or to modify it, and it leaves users and developers immense freedom to create structures of their choosing in accordance with their own visions and needs. Instead, it describes how neighbors must live together in the community, and what allowance they must make for each others' uses. A copyleft clause is like a reciprocal easement requiring homeowners to allow each other access to their yards; a source-sharing clause is like a requirement that all building plans be on file at city hall.

In other words, an open-source license provides a framework within which software developers can build software and a community together. The license is a crucial piece of shared infrastructure supporting the community, it must be designed to serve that function, and the community collectively must adopt a license which its members understand to serve that function effectively. This is an essential point of connection between online-commons theory and design theory.

---

58. *E.g.,* Neal Kumar Katyal, *Digital Architecture as Crime Control*, 112 Yale L.J. 2261 (2003); Eric J. Feigin, *Note: Architecture of Consent: Internet Protocols and Their Legal Implications*, 56 Stan. L. Rev. 901 (2004).

59. Or, to use his terms, the "code" of "cyberspace."

60. Michael Sorkin, Local Code:: The Constitution of a City at 42 degrees North Latitude (1996). Sorkin's use of "constitution" in his title is of course deliberate; he knew equally well what he was up to.

61. *See, e.g.,* Robert W. Gomulkiewicz, *General Public License 3.0: Hacking the Free Software Movement's Constitution*, 42 Hous. L. Rev. 1015 (2005) (discussing the metaphor and its sources).

Kelty, Frischmann, and others build on Ostrom by extending her work from tangible, physical infrastructure to informational goods. The communities these scholars study are responsible for commons in information. As they show, the governing institutions that these communities develop have a particular, recursive, character, and they manage the informational goods they are responsible for in ways that reflect its intangible freely shareable character.

Brand, on the other hand, focuses like Ostrom on the management of tangible, physical infrastructure. But his work too, deals with communities in an essential way. Both the high-road and low-road modes are responsive to the needs of the communities that use the buildings they maintain. High-road maintainers act as stewards and trustees for their communities, consciously repairing and reshaping buildings to a guiding vision of important community values. Low-road maintainers act in a more decentralized way, responding to immediate felt needs of community members.

2. The High Road and the Low Road in Open-Source Licensing

Connecting the traditions, for example, shows that the high road and the low road are not limited to physical infrastructure. These complementary modes of design and maintenance are visible in open-source development practice, as well. In his early and influential manifesto of open source, Eric Ramyond compared it to a chaotic "bazaar" in contrast to the carefully managed "cathedral" of proprietary software, arguing that this very organization looseness gave it far greater creative capacity.[62] The metaphorical contrast, of course, is between the stereotypical low-road building and the stereotypical high-road building. It transpired that many successful open-source projects are in fact run more like cathedrals, with a benevolent dictator or other central maintainer. But other projects are not, and the open-source ecosystem as a whole is a bazaar filled with cathedrals jumbled together with countless tiny and shifting shop stalls. In short, one advantage of open source development is precisely that it can adopt *both forms*; it can take the low road when the high road is blocked, or take the high road when the low road bogs down.

The simplest and most straightforward function of an open-source license is precisely to put software into a commons that is agnostic between development modes. A proprietary license locks development into the model preferred by the copyright owner. An open-source license allows developers to build cathedrals or bazaars, as they see fit, and as the needs of their user

62. Eric S. Raymond, The Cathedral and the Bazaar (1999). A more rigorous academic development of this theory is Yochai Benkler, *Coase's Penguin, or, Linux and The Nature of the Firm*', 2002 Yale L.J. 369.

and development communities demand. The specific licenses open-source projects adopt are tied to their development style. In turn, those modes are visible in the license drafting process itself.

Start with the high road. Some software projects are centrally maintained by a steward (a person or an organization) who takes responsibility for its development and long-term sustainability. Facebook, for example, largely supports open-source projects and open-sources much of its own work, following the high road approach.[63]

The fact that a high-road project is centrally maintained, however, does not mean that community is unimportant to it. It is precisely because they act as stewards that high-road maintainers must be vitally concerned with a community's needs. A project that is not may risk failure, through lack of adoption or lack of contribution. The choice to open-source a project *at all* is a decision about the role of the community in the project. It should not be surprising, then, that the stewards of high-road open-source projects often invite the relevant communities into the planning and deliberation process about their licensing decisions – and when they do not, community members often show up anyway, expecting to have a voice in these discussions.

Many low-road licenses are less obviously visible, precisely because they emerge from community adaptation rather than careful drafting. Indeed, this evolution results in many licenses that look awkward, even nonsensical, to the trained legal reader. But these licenses answer to the felt needs of their creators, and legal effectiveness is just one of several goals for a license.[64] Examples include:

- The Friends and Lovers License, which grants rights to "any person who has experienced mutual feelings of friendship or love with the author of this SOFTWARE at any time."[65]
- The Fuck Around and Find Out License, which provides that "the software shall be used for Good, not Evil. the original author of the software retains the sole and exclusive right to determine which uses are Good and which uses are Evil."[66]

63. Daphne Leprince-Ringuet, *Open Source at Facebook* (2021), https://www.zdnet.com/article/open-source-at-facebook-700-repositories-and-1-3-million-followers/.

64. *See generally An Anti-License Manifesto* (2021), https://www.boringcactus.com/2021/09/29/anti-license-manifesto.html (listing other unorthodox licenses).

65. *Friends and Lovers License* (2021), https://github.com/outofambit/friends-and-lovers-license.

66. *Fuck Around and Find Out License* (2020), https://git.sr.ht/~boringcactus/fafol/tree/master/LICENSE-v0.1.md.

- The Be Gay Do Crimes License, which is identical to the MIT License except that it adds the license condition to "Be Gay Do Crimes."[67]
- The Death and Repudiation License, which is based on the BSD License, but with provisions ensuring that the software "software may not be used directly by any living being."[68]

These licenses are inspired by each other; they propagate by mechanisms like those through which memes are continually remixed.[69] This is low-road evolution.

To conclude, then, contemporary open source licensing cannot be fully understood without seeing both the high road and the low road styles. Both the careful stewarding of licenses and community-driven licensing ferment are essential aspects of the story of open-source development.

II. Case Studies

The objectives of the open software and hardware movements are the same: to build technical and social ecosystems that center collaboration, sharing, and freedom rather than exclusion and restriction. These communities seek to foster an environment that encourages open exchange of ideas, feedback, and expertise, enabling participants to collectively overcome challenges and adapt products to meet diverse and individualized needs.

To ensure that this collaborative spirit endures and benefits all stakeholders involved, both movements operate within a legal framework based on open licensing. These licenses provide a legal guarantee specified individual freedoms in relation to software and hardware, and serve as the foundation upon which the principles of co-creation and sharing are enacted. They are carefully crafted to permit the widespread use, modification, and distribution of the software or hardware while upholding principles of openness, transparency, and community-driven development.

By embracing open licenses, the movements empower contributors to freely access, study, modify, and distribute the source code or hardware designs, ensuring that knowledge is not restricted, but rather shared and improved upon collectively. This open and inclusive approach cultivates a culture of continuous improvement, where the shared efforts of the community lead to enhanced and refined versions of the projects. The open-source soft-

67. *Be Gay Do Crimes License* (2021), https://github.com/Xe/waifud/blob/e7de416dbc0c14cf29e50b24e2d6337881294da9/LICENSE.

68. *Death and Repudiation License* (2003), https://github.com/indeyets/syck/blob/2656dcc9e879a26e0d7c36ae45f22150d2692ad0/COPYING#L26-L53.

69. *See generally* Limor Shifman, Memes in Digital Culture (2013.

ware and hardware movements do more than foster innovation; they also champion the ethos of democratizing technology and knowledge. These communities' commitment to providing accessible and adaptable solutions enables a broader spectrum of individuals and organizations to participate in the process of innovation, irrespective of financial or technical limitations.

This Part describes four collaborative projects in moments of growth, crisis, and change. Two of these case studies (GPL version 3 and the split between OpenOffice and LibreOffice) involve software; the other two (MakerBot and Arduino) involve hardware. To introduce these studies, we give a brief background on the history of the open software and hardware movements. In the case studies, we emphasize two themes: (1) the degree of community participation, (2) value articulation as an essential part of license design. We have chosen these case studies because they are moments when these two themes intersect.

A. Open Source Software

The development of computing infrastructure and hardware runs parallel to digital infrastructure or software development. Whereas computer hardware – motherboards, data storage, graphics cards, network adapters, etc. – constitutes the physical make-up of a computer system, software directs this hardware to carry out specified instructions.[70] Software programs regulate computer functionality, and general-purpose hardware has little to no value without it.

The United States legal framework for software copyright adheres to a vision most commonly associated with the 1978 report of the National Commission of New Technological Uses of Copyrighted Works (CONTU).[71] The report argued that the programmers invest significant effort to create economically valuable software, but the ease of digital copying threatens to undermine the incentive to make that investment. Thus, CONTU argued, copyright protection for software fit naturally into copyright's utilitarian structure of incentives, and software fits cleanly into copyright's conceptual structure as a "literary work" made up of symbolic tokens, like a poem or novel.

But this vision of software copyright has never been uncontroversial.[72] Both legal theorists and software developers have challenged it on philosoph-

70. Eben Moglen, *Anarchism Triumphant and the Death of Copyright*, 4 First Monday (1999).

71. Pamela Samuelson, *CONTU revisited: the case against copyright protection for computer programs in machine-readable form*, 1984 Duke L.J. 663.

72. Stephen Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs*, 84 Harv. L. Rev. 281 (1970).

ical and practical grounds. To them, software is functional first and foremost; software matters for what it does, not for what it says.[73] Wrapping software in copyright obstructs the practical ability of programmers to learn (uncopyrightable) ideas from other programmers' software, or to write programs that interact efficiently with it.

In contrast to CONTU's expectation of a commercial licensing market based on negotiated payments to use software, the free/libre/open-source (FLOSS) software community has established a radically different licensing framework. Proprietary licensing is typically built around "closed-source" programs, where the underlying source code is kept hidden from users. But in a FLOSS model, source code is widely available to the public and anyone is free to use, share, or modify it. The license explicitly guarantees these freedoms to users; the provision of source code makes those freedoms meaningful. The license secures negative liberty (from copyright's restrictions); the source code supplies positive liberty (to use and extend the software).

In addition, many open-source licenses, such as the General Public License (GPL), add a "copyleft" clause that allows the software to be published in a modified form only if the derivative works are also licensed under the GPL.[74] This ensures that software remains "free." The GPL uses "intellectual property rules to create a commons in cyberspace."[75] It initiates "a commons, to which anyone may add but from which no one may subtract."[76]

One project that was instrumental in the exponential expansion of the open source movement is the development of the Linux kernel which began as a side project in 1991 by Linus Torvalds to build a free operating system kernel. Its popularity and its organizational mythology, as captured in Eric S. Raymond's *The Cathedral and the Bazaar*, came to ultimately define key elements of open-source development.[77] Features of the Linux project include a large group of strangers arranged largely non-hierarchically who voluntarily collaborate on software over the internet. Working collaboratively in public became the basis of open source code as a "knowledge commons."[78] The key innovation of Linux

> *"...was not technical but sociological. Until the Linux development, everyone believed that any software as complex as an operating*

73. Moglen, *supra* note 70.

74. *GPLv3 Wiki* (2007), https://gplv3.fsf.org/wiki/index.php?title=Main_Page&printable=yes.

75. Moglen, *supra* note 70.

76. *Id.*

77. Raymond, *supra* note 62.

78. Charles M. Schweik & Robert C. English, Internet Success: A Study of Open-Source Software Commons (MIT Press 2012).

> *system had to be developed in a carefully coordinated way by a relatively small, tightly-knit group of people...*
>
> *Linux evolved in a completely different way. From nearly the beginning, it was rather casually hacked on by huge numbers of volunteers coordinating only through the Internet. Quality was maintained not by rigid standards or autocracy but by the naively simple strategy of releasing every week and getting feedback from hundreds of users.*"

When the Linux project switched to the GPLv2 license, it was adopted "*from the FSF, but [we believed] in it as an [engineering] choice and as a way to allow people to improve and share rather than as a moral imperative.*"[79] Unlike GNU which had a core team of developers who were physically proximate and could act as the "inside group" guiding development, Linux was truly made by strangers online. Linux needed the GPL, and the licensing regime informed the technical and sociological modes of the community's collaboration and growth.

1. Drafting GPLv3

The GNU General Public License (GPL) license aimed to provide a copyleft legal framework for distributing and using software in a way that aligned with the principles of the free software movement. First released in 1989, GPL quickly became one the most widely employed software license series in the world. The first revision of the GPL occurred in 1991 (GPLv2), when FSF added a new to respond to the growing issue of software patents and their potential impact on free software. Section 7 of GPLv2 specifically addressed the threat posed by patents by stating that if a licensee pursued legal action against someone else for patent infringement related to the software, their own license to use the software would be terminated. For example, if a company used GPL-licensed software and then attempted to sue another entity for patent infringement involving that software, they would lose their right to use the software under the terms of the GPL. Plainly put, Section 7 underscored that obligations of the GPL license cannot be severed to favor other conflicting obligations.

By 2006, continued expansion of the software ecosystem created numerous challenges for the community such as the scale of volunteer developers,

79. Tozzi, Christopher, *Torvalds Talks about Early Linux History, GPL License and Money: Open Source Application Software Companies content from The VAR Guy* (2016), https://web.archive.org/web/20170324170531/http://thevarguy.com/open-source-application-software-companies/torvalds-talks-about-early-linux-history-gpl-license-and-.

geographic expansion, and novel business and non-commercial uses. On the legal end, digital rights management (DRM) technologies and anti-circumvention laws further complicated the applicability and compliance of open software licenses. These ambiguities and omissions led to debates, legal uncertainties, and the need for an updated version of the license, culminating in the development of an organized GPLv3 revision campaign to address these language-related shortcomings and provide clearer guidance for the evolving challenges of the community.[80]

The revision process began in January 2006, when 350 participants convened at the Massachusetts Institute of Technology to kick-off the first of a long series of transnational public events.[81] The community was invited to participate in revision process the by either commenting on the public website, attending a conference, or participating in a discussion committee. Through the yearlong process, the community responded to the organizations' calls, and engaged extensively, and the following subsections focus on 2 issues which generated the most spirited discussions and debates on the GPLv3 online forum.

Tivoization

Tivoization refers to a practice in which manufacturers use digital rights management (DRM) or other technical measures to lock down their hardware devices, preventing users from modifying or running their own modified versions of the software running on those devices.[82] Notably, the term "Tivoization" is derived from the digital video recorder (DVR) company TiVo, which used this approach in its products.

In the context of open-source software, tivoization poses a challenge to the principles of software freedom and open collaboration. The former license did little to address this emerging issue.[83] Section 3.6 in GPLv2 states "*...you may not impose any further restrictions on the recipients' exercise of the rights granted herein*".[84] In response to this concern, the final draft of GPLv3 includes an explicit anti-tivoization provision:

80. *GPLv3 Wiki*, *supra* note 74. *See generally* Gomulkiewicz, *supra* note 61 (discussing revision process and key substantive debates).

81. *GPLv3 Wiki*, *supra* note 74.

82. John Tsai, *For Better or Worse: Introducing the GNU General Public License Version 3*, 23 Berkeley Tech. L.J. 547 (2008).

83. Brett Smith, *A Quick Guide to GPLv3*, Free Software Found. (2007), http://www.gnu.org/licenses/quick-guide-gplv3.html.

84. *The GNU General Public License v3.0*, *supra* note 18.

> "*If you convey an object code work under this section in, or with, or specifically for use in, a User Product... the Corresponding Source conveyed under this section must be accompanied by the Installation Information.*"[85]

This provision underwent refinement over the course of four drafts, with the FSF gradually building upon and improving the phrasing to address various concerns and intricacies.

The first draft of GPLv3 included a short, imprecise version of this provision under the "Non-Source Distribution" subsection. Overwhelmingly, users felt confused and questioned the provision's practical application: "*This clause seems far too vague and broad. It depends greatly on the interpretation of "users" and "immediate.*"[86]

In Draft 2 of GPLv3, a new subsection was created called "Conveying Non-Source" which stated that "*Distribution of the Corresponding Source in accord with this section must be in a format that is publicly documented, unencumbered by patents, and must require no special password or key for unpacking, reading or copying.*"[87] This version of the provision generated approximately 20 comments from the community, revealing further concerns and uncertainties regarding its scope and applicability:

One user asked:

> "*I agree with the goal of this paragraph. However, this places the burden on me to determine that the source format I am using is unencumbered by patents. How am I supposed to do this? Why should I be responsible for determining that eg gzip is unencumbered by patents? This seems like an unreasonable burden to me. It should be sufficient to distribute the source in a format that can be unpacked using tools distributed with the operating system.*"

[88]

By Draft 4, a small number of new comments emerged. One stated, "*This part is improved with respect to GPLv3draft3, as it no longer refers to U.S.-specific acts. Good.*" However, others still struggled to grasp the new section's purpose: "*This section seems to me to be too far reaching. The GPL is a software license mainly intended to cover software. In trying to extend its reach in*

85. *Id.*

86. *GNU GENERAL PUBLIC LICENSE: Discussion Draft 1 of Version 3, 16 Jan 2006* (2006), https://gplv3.fsf.org/comments/gplv3-draft-1.html.

87. *GNU GENERAL PUBLIC LICENSE: Discussion Draft 2 of Version 3, 27 July 2006* (2006), https://gplv3.fsf.org/comments/gplv3-draft-2.html.

88. *Id.*

this fashion to also impact hardware designs it is committing a mistake in my opinion..."[89]

The GPL is a preeminent example of a high-road license. The fact that it is so widely used gives it a special importance in the open-source world, and the FSF approached the revision process with caution and care. While the FSF took responsibility for final decisions about the license text, it sought and received extensive stakeholder input. The resulting text is a highly polished artifact; every word has been the subject of significant analysis and argument. It is the Hagia Sophia of licenses.

Digital Rights Management

The introduction of a new section within the GPL's provisions, focusing on anti-circumvention laws, digital rights management (DRM), and novel patent provisions, became another focal point of extensive debates and discussions. "*Protecting Users' Legal Rights From Anti-Circumvention Law*" (Section 3) was especially contentious and split the community between those who favored the ideals of openness and freedom as articulated by the FSF, and those who reasoned that OSI's perspective towards practical realities of software development and business were a more sustainable approach.[90] Section 3 first addresses "para-copyright" and states that all GPLv3 works are outside the scope of para-copyright measures:

> "*No covered work [under GPLv3] shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article II of the WIPO copyright treaty ... or similar laws prohibiting or restricting circumvention of such measures.*"[91]

The section continues with anti-DRM additions which make DRM and GPLv3 software legally incompatible. Patent related provisions were also added, restricting those who "convey" GPLv3 works from suing or asserting patent rights against third parties.[92] Criticism towards section 3 can be summarized by Linus Torvalds' concerns that the anti-DRM section is "*...a clear example of a choice being made on the 'religious' tenets of the FSF rather*

89. *GNU GENERAL PUBLIC LICENSE: Discussion Draft 4 of Version 3, 31 May 2007* (2007), https://gplv3.fsf.org/comments/gplv3-draft-4.html.

90. Clark D. Asay, *The General Public License Version 3.0: Making or Breaking the FOSS Movement*, 14 Mich. Telecomms. & Tech. L. Rev. 265—301 (2008), https://repository.law.umich.edu/mttlr/vol14/iss2/1.

91. *Id.*

92. *Id.*

than the appropriate technical grounds."[93] While Torvalds had been a strong proponent of the GPL, his concerns (and that of many others in the community) aligned with the OSI's perspective.[94]

Online, the new DRM provision in first draft of GPLv3 generated dozens of comments. Community members raised a variety of concerns regarding the provision related to privacy-invading actions and patent licenses in the GPLv3 draft.[95] Some expressed that the language of the provision could be ambiguous and not sufficiently clear, making it challenging to discern its intent:"*I find the entire section 3 to be inscrutable. The rest of GPL is in plain English, but this is lawyer talk.*" Another user worried that, "*...this may be a restriction on use of the works disguised as a restriction on distribution.*"[96]

Others questioned the need for this provision within the GPL, while some worried that it might unintentionally restrict certain legitimate uses of GPL-licensed software, such as security tools. There were debates about the scope and potential impact of the patent license grant, with suggestions to clarify whether it covered claims related to downstream modifications and practices beyond software. The concern was that while the intent was clear, the wording and potential loopholes might not align with the desired outcomes. For example, "*In order to make sure that people don't use this loophole, a new sentence should be added stating that the copyright holders grant licenses over any patent claims they might hold over the program they distributed (but not necessarily over patent infringements added downstream?)*"[97]

There were also discussions about the interplay between this provision and other sections of the GPL, such as its effect on distributors under different versions of the license. By the fourth draft and under a newly defined subsection, users still expressed some reservations: "*This clause is clearer than in the previous draft, but still troublesome, as it seems to be overreaching. For instance, it could be interpreted as covering legal powers to forbid "computer crimes" such as unauthorized intrusion into computer systems.*"[98] Despite this reservation, the language in draft 4 was not altered again, suggesting a measure of consensus within the community regarding its formulation. Further, engagement on this specific provision in the online forum appeared to di-

93. Steven J. Vaughn-Nichols„ *Linux-Watch: Is GPL 3 Dead on Arrival?* (2006), https://www.linuxtoday.com/infrastructure/linux-watch-is-gpl-3-dead-on-arrival/.

94. Asay, *supra* note 90.

95. *GNU GENERAL PUBLIC LICENSE: Discussion Draft 1 of Version 3, 16 Jan 2006*, *supra* note 86.

96. *Id.*

97. *GNU GENERAL PUBLIC LICENSE: Discussion Draft 2 of Version 3, 27 July 2006*, *supra* note 87.

98. *GNU GENERAL PUBLIC LICENSE: Discussion Draft 4 of Version 3, 31 May 2007*, *supra* note 89.

minish by the fourth draft. This decline in discussions and feedback could be indicative of a growing alignment among community members around this particular clause.

After 4 drafts, 2635 comments, and 18 public events across 12 countries, the FSF finally released the final version of GPLv3 in 2007.[99] The GPLv3 is a testament to the community's self-regulatory capacity as well as its dedication to promoting knowledge sharing and centering the principles of free software development. The GPLv3 revision process showcases the highly collaborative nature of license drafting, reflecting the community's commitment to centering shared objectives in the evolving landscape of software development. For the FSF, copyright is infrastructure that holds the community together. In other words, without the software production community, the license is meaningless. The community, therefore, were the most important stakeholders situated at the core of the revision process and each iteration of the draft responded to community concerns, questions, perspectives, and signals.

2. OpenOffice and LibreOffice Split

OpenOffice and LibreOffice are open source office suites. Both are free and both provide features for word and text processing similar to proprietary software like Microsoft Office. The two software suites were not always separate entities. Originally created in 2000 as a fork of the proprietary office suite *StarOffice*, SunMicrosystems (Sun) acquired the renamed *OpenOffice* project and released it as open source software.[100] During this period, the *Community Council*, comprised of members of the OpenOffice community and volunteers governed the project.[101] The Community Council worked alongside Sun and Oracle to centralize project goals and tasks, and coordinated with derivative projects. However, even early on, OpenOffice's governance model was fraught conflicting interests and power structures. Both Sun and Oracle authorized decisions without the Community Council's approval, or sometimes, against their recommendation.[102]

Although the OpenOffice project scaled and popularized quickly, once Sun was acquired by Oracle in 2010, the organization moved to terminate

99. *GPLv3 Wiki*, *supra* note 74.

100. Ankush Das, *LibreOffice vs OpenOffice: All You Need to Know?* (2022), https://itsfoss.com/libreoffice-vs-openoffice/.

101. *Oracle wants LibreOffice members to leave OOo council* (2010), https://web.archive.org/web/20120625041345/http://arstechnica.com/information-technology/2010/10/oracle-wants-libreoffice-members-to-leave-ooo-council/.

102. *Community Council Charter* (2009), https://web.archive.org/web/20110424032526/http://council.openoffice.org/councilcharter12.html.

commercial development of the project.[103] In the following year, Oracle attempted to showcase their commitment to the broader open source community by donating all OpenOffice code to the Apache Software Foundation.[104] This step, however, initiated a serious of licensing changes.

Prior to the acquisition, the OpenOffice suite had been released under the GNU Lesser General Public License (LGPL). Following the acquisition, the OpenOffice.org project switched to the non-copyleft Apache License.[105] This meant that while all prior versions of OpenOffice would remain under the LGPL, all future releases would be licenced under Apache 2.0.

In parallel to OpenOffice's evolution, is the story of LibreOffice. Towards the goal of maintaining a truly *open* project following Oracle's acquisition of SunMicrosystems and the forcible expulsion of developers from the Community Council, the community created The Document Foundation (DF). Under a new, non-profit infrastructure, the Document Foundation sought to support the development of open source code for the office suite that aligned with values of the broader community.[106] The new project was newly named *LibreOffice*, and at first, the software was a fork of all the existing OpenOffice code and a large sum of volunteers committed to the project's long-term evolution and sustainability. In the early days of LibreOffice, volunteers viewed it as *"...a healthy, active community project, while OpenOffice is an abandoned corporate zombie. It is on life support living off its trade mark name."*[107]

For the community, splitting from Sun meant that the LibreOffice community had autonomy to differentiate itself from prior leadership, and discussions within the community mainly focused on (1) how the developer experience could be enhanced, and (2) how copyright would be assigned. On the governance side, Oracle pushed out all of OpenOffice's Community Council member's involved or sympathetic to LibreOffice.

Discussions on the LibreOffice forum signal appreciation towards the communal approach taken by the new organization:

> *"LibreOffice community has put a huge amount of work into improving their developer experience - speeding up builds, improv-*

103. Al Williams, Hackaday, *OpenOffice Or LibreOffice? A Star Is Torn* (2020), https://hackaday.com/2020/11/02/openoffice-or-libreoffice-a-star-is-torn/.

104. *Released: Apache OpenOffice 4.1.14* (2005), https://www.openoffice.org/FAQs/license-change.html.

105. Williams, *supra* note 103.

106. David Bretthauer, Open Source Software: A History (2001) (unpublished manuscript), https://opencommons.uconn.edu/libr_pubs/7.

107. *How to choose between LibreOffice and Apache OpenOffice?* (2014), https://www.reddit.com / r / libreoffice / comments / 2q6vt7 / how_to_choose_between_libreoffice_and_apache/.

> *ing infrastructure, refactoring code to make it more readable... it's interesting to see that those efforts are leading to more developers, increasing rate of change, and hopefully to improving market share.*"[108]

Related to copyright, discussions were more political due to the complex history of OpenOffice. Under Sun, for example, OpenOffice contributors were required to assign copyright over the company. This meant that Sun had control over what contributions were accepted and rejected, but more importantly, "*copyright reassignment is an anathema to open source collaboration*."[109]

On the one hand, OpenOffice's Apache License 2.0 license enabled easy integration with proprietary software. However, the licences permissive nature comes with some limitations in terms of code sharing. While Apache 2.0 allows other projects to use code from OpenOffice, it does not require modifications or derivative works to be released under the same license. This means that while OpenOffice can be integrated into other projects, those projects are not obligated to share their improvements back with the OpenOffice community.

LibreOffice, on the other hand, adopted a dual licensing approach combining LGPLv3 and MPL to re-center their copyleft commitments. When code is incorporated into LibreOffice under the LGPLv3, any modifications or derivative works must also be released under the LGPLv3 or a compatible license. This promotes code sharing and ensures that improvements to LibreOffice remain open-source. For example, the Apache 2.0 license states:

> *"You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions: You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work [...].*"[110]

Conversely, the Mozilla Public License 2.0 requires:

---

108. Jonathan Corbet, *Development activity in LibreOffice and OpenOffice* (2015), https://lwn.net/Articles/637735/.

109. Scott Merrill, *LibreOffice and OpenOffice.Org: One Year After the Schism*, TechCrunch (2011), https://techcrunch.com/2011/10/07/libreoffice-and-openoffice-org-one-year-after-the-schism/.

110. *Apache License, Version 2.0* (2004), https://www.apache.org/licenses/LICENSE-2.0.

> *"The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims [...] to use, reproduce, modify, display, perform, sublicense and distribute the Original Code [...] You may distribute Covered Software in Executable form [...] provided that You also meet all of these conditions: If You distribute any portion of the software in Source Code form, You may do so only under this License by including a complete copy of this License with Your distribution*."[111]

These licensing choices impact project governance and development models. The Apache License 2.0 used by OpenOffice takes a more permissive approach, enabling corporate entities to contribute code without requiring them to share their proprietary enhancements. This approach supports a development model that may invite a broader range of contributors, including corporations that do not contribute back to the main project.

These change prompted strong reactions from the FSF community:

> *"All Apache projects are distributed under the terms of the Apache License. This is a non-copyleft free software license; anybody who receives the software can distribute it to others under non-free terms. Such a licensing strategy represents a significant policy change for OpenOffice.org. Previously, the software was distributed under the terms of the GNU Lesser General Public License (LGPL)...*"[112]

And reactions continued across popular open source news sites (e.g. LWN.net) with perspectives from the broader open source community carefully following the split. One user opined that "*Politically, the general perception appears to be that Apache is operating as a convenient foil for IBM to usurp the OpenOffice brand name and community*."[113] Another participant in the same thread reacted to a comment on smooth community transition, arguing that

> *"An overlap in the communities does not prove that the communities are in agreement. Plainly some significant contributors of LibreOffice code feel that the present situation may make it impossible to successfully rebase LibreOffice on the Oracle code dump.*

111. *Mozilla Public License, version 2.0* (2010), https://www.mozilla.org/en-US/MPL/2.0/.

112. Brett Smith, *Statement on OpenOffice.Org's Move to Apache — Free Software Foundation — Working Together for Free Software* (2011), https://www.fsf.org/news/openoffice-apache-libreoffice.

113. Jake Edge, *Development activity in LibreOffice and OpenOffice* (2012), https://lwn.net/Articles/495627/.

> *That others have failed to register such concerns does not mean that they do not exist*."[114]

These sentiments point to the ongoing debates surrounding licensing choices and their direct implications on the openness and freedom of software. In other words, they highlight the dynamism of the open-source landscape where communities need to balance pragmatism and openness, while preserving the core principles of free software.

The dual licensing approach of LGPLv3 and MPL adopted by LibreOffice emphasizes the principle of reciprocity. Since modifications must be released under compatible licenses, the LibreOffice community collectively centers a community-centered development model. In doing so, contributions are more seamlessly shared back with the community, creating a stronger sense of collaboration, transparency, and long-term project sustainability. OpenOffice's transition to a non-free license created barriers for the community supporting the project, barriers that did not align with the community's goals for the project. One participant's comments that these barriers have stymied progress at OpenOffice:

> *"In the 4½ years since its founding, the LibreOffice project has put together a community with over 250 active developers. There is support from multiple companies and an impressive rate of patches going into the project's repository. The project's ability to sustain nearly monthly releases on two branches is a direct result of that community's work...it seems clear that the project is on a solid footing with a healthy community. OpenOffice, instead, is driven by four developers from a single company — a company that appears to have been deemphasizing OpenOffice work for some time. As a result, the project's commit rate is a fraction of what LibreOffice is able to sustain and releases are relatively rare.*"[115]

Sun's development of OpenOffice was another classic example of high-road maintenance. Sun accepted and rejected contributions to the official main branch of OpenOffice, and it also required copyright assignments for all contributions. Notice the coupling between Sun's *technical* stewardship over the code and its *legal* stewardship over the copyright. Contributors gave their code to the community by way of Sun, trusting it to make healthy decisions for the project. This kind of concentration of copyright ownership is a high-road technique; it empowers the steward to take legal action to enforce license terms. By contrast, when open-source codebases are frequently

114. *Id.*
115. Corbet, *supra* note 108.

forked and remixed, that is the low road in action. The code evolves through a process of branching and remixing that is intentionally facilitated by open-source licensing.

Oracle's acquisition of Sun threw a monkey wrench into Sun's high-road stewardship of OpenOffice because Oracle discontinued commercial development of OpenOffice, abdicating its role as steward. Community members would have been powerless to prevent a physical building from falling into disrepair or being converted into condominiums. But they could – and did – fork OpenOffice as LibreOffice. The choice to switch from the non-copyleft Apache license back to the copyleft LGPL was a deliberate choice to change the nature of the legal relations among community members.

B. Open Hardware

The open hardware community emerged in the early 2000s in response to the increasingly closed nature of the hardware industry. It sought to apply the principles of freedom and openness championed in the open software movements to hardware. But there are substantial differences between software and hardware that posed distinctive practical and licensing challenges.

Hardware projects have two categories of outputs: (1) design documentation (i.e. instructions to be followed by people or by machines), and (2) manufactured products.[116] This division follows the traditional division of software development into human-readable source code and executable computer code. But immediately it is clear that hardware is different. Whereas the copyright analysis of source and object code is substantially similar, open hardware relies on different legal frameworks for protecting documentation, products, and various elements and features in the design process. Copyright applies differently to virtual designs and the physical products built from those designs.[117] Design patents are more readily available for physical products than for software or screen displays, and utility patents face fewer doctrinal obstacles for hardware than for software.[118]

Hardware production can encompass many distinct and independent steps, depending on the product. Electronic hardware products like circuit boards, for example, require a schematic diagram to visually represent a circuit's components and wiring. A developer turning the schematic diagram

116. John R. Ackermann, *Toward Open Source Hardware*, 34 U. Dayton L. Rev. 193—222 (2009).

117. 17 U.S.C. § 113; James Grimmelmann, *Indistinguishable from Magic: A Wizard's Guide to Copyright and 3D Printing*, 71 Wash. & Lee L. Rev. 683 (2014).

118. *See* Ackermann, *supra* note 116 (discussing limitations of utility patents for open designs).

into a physical circuit board must make numerous design decisions about how to lay out the board. While the Copyright Act protects "pictorial, graphic, and sculptural works,"[119] it does not protect those that serve a utilitarian function (like a circuit board) unless the utilitarian function can be separated.[120] To track this difference, and others, open hardware licenses must use different license language and a different conceptual vocabulary.[121] For a time, Creative Commons (CC) licenses were the *de facto* open licensing suite for open hardware projects and components. However, CC licenses primarily apply to source files and documentation, and do not apply cleanly to the hardware itself.

Thus, following the model of free and open software developers, open hardware innovators and communities crafted hardware-specific licenses. In 2007, the Tucson Amateur Packet Radio (TAPR) project released the Open Hardware License (OHL), a copyleft license which facilitates the distribution and modification of hardware designs and requires that derivative works be released under the same license terms.[122] It also includes a lengthy preamble to guide and prepare their highly technical audience:

> *"Unlike the GPL, the OHL is not primarily a copyright license. While copyright protects documentation from unauthorized copying, modification, and distribution, it has little to do with your right to make, distribute, or use a product based on that documentation. For better or worse, patents play a significant role in those activities…*
>
> *The OHL addresses unique issues involved in the creation of tangible, physical things, but does not cover software, firmware, or code loaded into programmable devices. A copyright-oriented license such as the GPL better suits these creations."*[123]

In 2011, the European Organization for Nuclear Research (CERN) developed another open hardware license.[124] Like OHL, the CERN license is also copyleft and requires modified designs to be made available under the same license terms, with the additional requirement that the license be included

119. 17 U.S.C. § 102(a)(5).

120. Star Athletica, LLC v. Varsity Brands, 137 S.Ct. 1002 (2017).

121. Ackermann, *supra* note 116.

122. *The TAPR Open Hardware License* (2007), https://tapr.org/the-tapr-open-hardware-license/; Andrew Katz, *Towards a Functional License for Open Hardware*, 4 Int'l Free & Open Source Software L. Rev. 41 (2012).

123. *The TAPR Open Hardware License*, *supra* note 122.

124. Myriam Ayass & Javier Serrano, *The CERN Open Hardware License*, 2012 HeinOnline 71.

in all distributions of the design as well.[125] CERN also adds provisions for patents and liability to further protect contributors.[126]

Existing organizations like TAPR and CERN that created open licenses were soon joined by an organization dedicated to open hardware licensing: the Open Source Hardware Association (OSHWA).[127] Founded in 2012, OSHWA's main objective is to promote the adoption of open hardware principles. While OSHWA did not participate in the development of the early open hardware licenses, OSHWA played a crucial role in educating the broader open-hardware community. In addition to centralizing the OH community, OSHWA developed the "Open Hardware Certification Program" as an educational tool to help developers understand the intellectual property issues associated with designs and projects.[128] The online-certification program enables developers to self-certify their hardware designs as OSHWA-compliant open source hardware. To be eligible for certification, projects must comply with OSHWA's defined criteria for openness, transparency of design and documentation, and licensing and distribution practices. OSHWA also participates in numerous initiatives that promote the growth and development of the open hardware community.[129] These include organizing conferences and events, publishing educational resources and guidelines, and advocating for policies that support open-source hardware development. In more ways than one, OSHWA is interested in creating a social norm for adopting OH licenses and certifications. Presently, OSHWA recognizes both the TAPR and CERN open hardware licenses, and maintains a robust list of open hardware licenses on its website.[130]

None of this has been conflict-free. Early examples included disptutes over the definition of "open hardware" itself and whether to include commerciality clauses in open-hardware licenses.[131] Another well-known example is debate within the community concerning the openness of the layout of a printed circuit board. Stakeholders associated with the open hardware and software company Arduino advocated that files of printed circuit board layout should be treated as a trade secret. They argued that the layout of a board can be contain artistic expression, and that circuits that could be laid out in different ways. This argument was ultimately outvoted, and open-hardware licenses generally require the sharing of layout files.

---

125. *Id.*
126. Katz, *supra* note 122.
127. *Open Source Hardware Association* (2023), https://www.oshwa.org/.
128. *Id.*
129. *Id.*
130. *Id.*
131. Citations t/k

1. The Closing of Makerbot

MakerBot was a celebrated favorite of the open hardware community. Founded in 2009, the team behind Makerbot was on a mission to engineer accessible and modifiable 3D printers for the masses. The first MakerBot model, the Cupcake Computer Numerical Control (CNC), was a do-it-yourself (DIY) 3D printer kit.[132] The Cupcake CNC, and Makerbot as an organization, were highly influenced by the RepRap project, an initiative focused on creating open-source 3D printers capable of self-replication, where parts of the printer could be 3D printed by the printer itself.[133] At the time, RepRap was where one would find 3D printing enthusiasts and Makerbot launched the Cupcake CNC alongside a clearly articulated alignment with RepRap's open-source design philosophies which helped maintain community continuity and advancement within the growing 3D printing landscape.

The growing online 3D printing ecosystem thrived on collective efforts to refine the printer's design, exchange innovative modifications, and co-create software advancements. Guided by RepRap, the 3D printing community had already established norms of sharing: publishing designs to open online repositories under Creative Commons licenses. Specific to this community of creators, CC licenses define the terms under which designs can be used, modified, and distributed. In the context of 3D printing, these licenses safeguard the digital design files, typically in STL format[134], which serve as the blueprints for creating physical objects. By applying a CC license to an STL file, creators can choose whether they want to share their design openly, require attribution for its use, allow or restrict commercial use, and even mandate that any derived works are shared under the same license.

Following in RepRap's footsteps, MakerBot released its early project (specifically, the Cupcake CNC in 2009, and the Thing-O-Matic in 2010) designs under Creative Commons licenses and established a platform for sharing open design files. Thingiverse, an online platform where users could share and collaborate on 3D models, designs, and projects became a central hub for the 3D printing community to freely exchange ideas, designs, and modifications.

In 2012, MakerBot released the Replicator which represented the most significant technical milestone for the company, and for the open 3D printing community yet. The Replicator offered a dual extruder setup, which enabled multi-material and multi-color printing. This model marked a no-

132. Robin P. G. Tech, Jan-Peter Ferdinand & Martina Dopfer, *Open Source Hardware Startups and Their Communities: The Case of 3D Printing*, 2016 Springer 131.

133. *Id.*

134. 3D print design files, called STL (short for stereolithography)

table evolution in design and capabilities, and it was one of the last models from MakerBot to be released with open-source components. In their early more community focused years, MakerBot demonstrated its commitment towards preserving and championing open norm of creation and sharing. However, when a re-imagined version of the MakerBot printer appeared on Kickstarter,[135] MakerBot felt threatened by commercial markets. By this point, MakerBot had also gained traction and attracted investment and the company's attitude toward open source began to shift.

Later in 2012, Makerbot released the *Replicator 2* which shocked the community with its composition of proprietary hardware and software. While MakerBot continued to release certain software components under open-source licenses (such as parts of the ReplicatorG software), the hardware designs and core software of the Replicator 2 were proprietary. The company's leadership believed that a more controlled approach was necessary to maintain quality and protect intellectual property.

This decision was met with significant criticism from the open 3D printing community, who saw Makerbot's shift as a betrayal of the company's open source principles. Further, community grievances about MakerBot's licensing decisions raised questions about the role of open source principles in commercial ventures. Critics argued that MakerBot's decision to abandon open source licenses violated the trust of the community that had contributed to its early success. Others held the opinion that MakerBot's decision was a necessary step for the company to protect its intellectual property and remain competitive in the growing 3D printing market.

Innovations within the open 3D printing community come in the form of design files co-produced and widely distributed and utilized by the active members and other novice users. Within the expansive and distributed network of designers, engineers, hobbyists, manufacturers, and volunteers within the community, it is difficult to locate where the innovation actually happened, or who is responsible. Within open hardware communities, a completed design can also hold sentimental value, where owning a design or product fits into the notion of owning an idea - a notion the open hardware community, particularly the 3D printing community, labored to work against. This tension was further fueled by the sale of MakerBot to Stratasys in 2013. Stratasys took a corporate approach and quickly filed patents for many of the design features of the printers such as the three-dimensional printer with force detection and the quick-release extruder.[136] Members of

135. Brian Benchoff, Hackaday, *The MakerBot Obituary* (2016), https://hackaday.com/2016/04/28/the-makerbot-obituary/.

136. Three-Dimensional Printer with Force Detection, U.S. Patent No. 9,168,698 (issued 2014–10–27).

the community felt that MakerBot's decision to patent technologies developed through open collaboration was hypocritical and undermined the ethos of the community. As one participant argued,

> *"A patent is definitely a legal term restricting what you can do with the work. It violates the CC license, which means either any of the contributors could theoretically revoke makerbots permission to use it. In actuality, makerbot has a large legal budget, and their terms of service probably say something about providing a license to the content anyway. They've used semi-legal trickery to steal thingiverse users' designs, and without a large legal fund there isn't really any recourse...*
>
> *Makerbot has shown time and time again that they're willing to screw over the community. Are we really that surprised? But their marketing budget and sleek design continues to convince the uninformed to buy makerbot. To add insult to injury, their printer is overpriced, has reliability problems, and that compounds with sub par support."*[137]

MakerBot leadership responded to these moments of crises articulated by their community, but the public had already been damaged. Tensions over MakerBot's licensing decisions underscore the need for clear and transparent licensing arrangements in addition to governance structures in open source projects. In the absence of clear copyright protections for open hardware innovations, how can a project or company balance the needs of different stakeholders, while preserving open source ideals and the intellectual property of open source designs released to the Commons? While the open software cases describe successful moments of enculturation, legal pedagogy, and crisis management,[138] the open 3D printing community demonstrates that licensing infrastructure cannot always adjust. Instead, these ethical and political moments become tools through which the community learns something about their shared identity. In the case of MakerBot, the community collectively evacuates infrastructure that does not serve their objectives. They re-organize elsewhere, within a different pre-existing but adaptable structure, or create a new structure entirely.

The MakerBot fiasco is another example of failed stewardship. The shift to proprietary hardware and patent protections undercut the bargain that

137. Traverseda, *Makerbot Blatantly Steals and Patents a Community Design.*, INDISTINGUISHABLE FROM SCIENCE (2014), https://traverseda.wordpress.com/2014/05/23/makerbot-blatently-steals-and-patents-a-community-design/.

138. Coleman, *supra* note 8.

community members believed the community to be based on. It was an expensive renovation that turned a bustling marketplace of small vendors into a sterile big-box store. MakerBot may have been right that it could not afford to continue the original Replicator model, but the revised legal design it chose was incompatible with the purposes that model served for community members.

2. The Arduino Trademark Tussle

In 2005, a group of students at the Interaction Design Institute Ivrea in Italy created an accessible and user-friendly platform that would enable individuals, particularly those without specialized technical knowledge, to experiment with electronics and microcontrollers.[139] The developers centered open-source principles that would become the foundation of the project's philosophy, and since its inception, Arduino has become a cornerstone of the Maker Movement, empowering individuals to bring their creative ideas to life in the realm of electronics and technology.

The team, led by Massimo Banzi, developed the *Arduino* board, based on microcontrollers that can be programmed using the Arduino Integrated Development Environment (IDE). The boards come in various shapes and sizes and are equipped with digital and analog input/output pins, enabling connections to sensors, actuators, and other electronic components.[140] Arduino's versatility, combined with an international community of developers and readily available libraries, makes it ideal for prototyping and experimenting with a wide range of projects, from robotics and automation to home automation and wearable devices.

The core software libraries and IDE are licensed under the GPL.[141] Releasing under the GPL ensures that any modifications or enhancements made to the Arduino software must be shared under the same license, promoting code reciprocity and collaborative development within the community. Arduino's hardware designs (including the designs of various Arduino boards like the Uno and Mega) are released under the Creative Commons Attribution-ShareAlike (CC BY-SA) license.[142] CC allows for the free use, modification, and distribution of the hardware designs as long as the original creators are credited, and any derivative works are shared under the same license. A dual-licencing approach encourages hardware manufacturers to produce

139. *The History of Arduino Part 1* (2023), https://circuitdigest.com/article/history-of-arduino-part-1.

140. Yusuf Abdullahi Badamasi, *The Working Principle of An Arduino*, 2014 IEEE 1.

141. *The History of Arduino Part 1*, *supra* note 139.

142. *Id.*

Arduino-compatible boards, which has led to a proliferation of options and variations. At the same time, it ensures that the fundamental design remains open and accessible, so the community can continue to innovate, adapt, and create new hardware based on the original Arduino concepts.

Arduino's open-source philosophy extends to third-party libraries and components developed by the community.[143] Third-party libraries and components play a crucial role in extending the functionality and versatility of the Arduino platform. These libraries are typically created and maintained by developers outside of the core Arduino team, and they are shared with the broader community to enhance the capabilities of Arduino boards.[144] Third-party libraries and components cover a wide range of functions and applications including specialized sensors, communication protocols (like Wi-Fi or Bluetooth), displays, motor control, among others. Broadly speaking, libraries are developed to simplify complex tasks and enable users to easily integrate various hardware and software features into their projects.[145] To enhance access and discoverability, the Arduino IDE includes a Library Manager tool that simplifies the process of discovering, installing, and updating third-party libraries. Users can search for libraries within the IDE, download and install them with a click, and keep them up to date easily. This feature in Arduino's IDE streamlines the integration of new functionality into Arduino projects. Many of the libraries and add-ons that expand the functionality of Arduino are also released under open-source licenses. The specific licenses for these libraries and components can vary, but many adhere to similar open-source principles, such as the GNU GPL or various Creative Commons licenses.[146]

For example, the ESP8266 Core is a critical component within the Arduino ecosystem that enables the programming and utilization of ESP8266 microcontrollers.[147] Importantly, microcontrollers are known for their low cost and integrated Wi-Fi capabilities. The ESP8266 core, therefore, provides a set of tools, libraries, and frameworks that simplify the development process for ESP8266-based projects, making it accessible to a wide range of users, from hobbyists to professional developers. Its significance lies in its ability to harness the power of the ESP8266, allowing the creation of Internet of Things (IoT) devices, smart home solutions, and numerous other applications that rely on Wi-Fi connectivity. With the ESP8266 Core, Arduino extends its

143. *Licensing for products based on Arduino* (2023), https://support.arduino.cc/hc/en-us/articles/4415094490770-Licensing-for-products-based-on-Arduino.

144. Badamasi, *supra* note 140.

145. *Id.*

146. *Id.*

147. *Arduino esp8266* (2020), https://github.com/esp8266/Arduino/.

compatibility to a broader range of hardware, as well as individuals and communities, to build connected and intelligent devices and contributing to the growth and diversity of the Arduino ecosystem.[148]

The ESP8266 package is comprised of a number of bundled libraries that released under different licences.[149] The diverse licensing of components within the ESP8266 Core highlights the collaboration and integration of various tools and libraries from different sources to create a comprehensive development environment for ESP8266 microcontrollers (see table X). Each component serves a specific purpose, and the choice of license reflects the ideals and requirements of its original authors.[150]

Component	**License**	**Description**
Arduino IDE	GNU General Public License (GPL)	Software environment for programming Arduino boards.
ESP8266 Core (xtensa gcc toolchain)	GPL	Toolchain used for compiling code for ESP8266 microcontrollers.
Esptool	GPLv2	Tool for flashing firmware onto the ESP8266.
Espressif SDK	Espressif MIT License	SDK included in this build for the ESP8266, specific to Espressif Systems.
ESP8266 core files	GNU Lesser General Public License (LGPL)	Core files for programming the ESP8266.
SPI Flash File System (SPIFFS)	MIT License	File system library.
ummmalloc Memory Management Library	MIT License	Memory management library.
axTLS Library	BSD License	Library used in this project, built from the GitHub repository.

For instance, the inclusion of the *xtensa gcc toolchain* and the *Esptool* under the GPL aligns them with the principles of the broader Arduino platform which enables unrestricted modification and distribution. The *Espressif SDK*, licensed under the Espressif MIT License, is specific to the manufacturer of the ESP8266, and likely includes proprietary elements or features closely tied to the hardware. Finally, licensing the *ESP8266 core files* under the GNU Lesser General Public License (LGPL) provides the flexibility to

148. *Id.*
149. *Software Licensing?* (2018), https://forum.arduino.cc/t/software-licensing/525281/2.
150. *Id.*

link with non-GPL code, allowing for a broader compatibility and integration of ESP8266 microcontrollers into various projects. Once again, a diverse licensing approach underscores the importance of creating a versatile and inclusive development environment that simultaneously respects the principles and legal requirements of each component's original creators. In doing so, it allows for the seamless integration of these tools into the Arduino ecosystem, enabling the community to leverage the ESP8266's capabilities for a wide array of applications.

As Arduino gained popularity tensions regarding the ownership of the Arduino trademark emerged, leading to a legal dispute between two companies, Arduino LLC and Arduino SRL.[151] On the one hand, Arduino LLC, founded by the original Arduino team in 2009, ran the website arduino.cc and directed the development and release of Arduino's code. Arduino SRL, on the other hand, formerly known as Smart Projects SRL, had been a major producer of Arduino boards since the project formalized, and later registered the domain arduino.org.[152] The conflict began when Arduino SRL began selling Arduino boards under the Arduino name, leading to a trademark infringement lawsuit filed by Arduino LLC.[153]

The trademark disputes caused a split within the Arduino community. Both factions released separate versions of the Arduino IDE, creating chaos and confusion within the community that also affected Arduino resellers. Users, specifically, were uncertain about which version of Arduino to trust as both companies continued to manufacture and sell Arduino boards under identical names.[154]

Across the Arduino Forum, countless community members engaged in lengthy debates and deliberations. One community member candidly raised concerns about the current state of licensing and the potential consequences of commercializing projects created with Arduino setups:

> *"But with the current state of ambiguity in the licensing, I really dont know what could happen if I sold a work that used an arduino setup that I made, since I would then be making commercial gain in a sense. Now yeah its not like Im going to get rich making*

151. Brian Benchoff, *Arduino Vs. Arduino: Arduino Won* (2016), https://hackaday.com/2016/10/01/arduino-vs-arduino-arduino-won/.

152. Elliot Williams, *Arduino V Arduino: Part II* (2015), https://hackaday.com/2015/03/12/arduino-v-arduino-part-ii/.

153. Nathan Willis, *A Trademark Battle in the Arduino Community* (2015), https://lwn.net/Articles/637755/.

154. *Id.*

> *my little arduino ripoffs but all the same these are the types of issues that need to be addressed as the Arduino grows.*"[155]

Another user tried to answer questions emerging across the forum regarding the ongoing battle that unfolded between Arduino.cc and Arduino.org—a distinction rooted in differing claims to the mantle of the "true" Arduino. This internal discord materialized as a microcosm of the broader fight for brand identity and trademarks.

> *"It is part of the on-going fight between Arduino.cc (here), and Arduino.org (who has split off and claims to be the "true" Arduino. Not many folks on here agree with that stance). Genuino is apparently a new name for Europe to address an issue with who holds the Arduino trademark in Europe (at least that is my understanding. Someone correct me if I got that wrong please).*"[156]

For more context, the term "Genuino" emerged as a novel identifier of Arduino tailored for the European community. Other voices across the Forum underscored the nuances of the open-source philosophy and its implications for ownership and governance.[157] For example:

> *"...The OSI definition does not discriminate against field of endeavour... I think the Arudino "business" (i.e. the files, design, name etc) should belong to anyone who wants to participate, and not be controlled by a single entity. This is how the project got so many supporters, by advertising itself implicitly as doing that. Given the fact that the community put so much into the project and made it successful as a result, I think it's reasonable to say that community owns "the Arduino business", not the team.*"[158]

This sentiment advocating for communal ownership of the broader Arduino "business" are archived across countless threads on the Arudino Forum. Community members reasoned that Arduino's trajectory was imprinted by a multitude of contributors who propelled Arduino through active participation and labour in the form of articles, blog posts, and sizeable projects.[159]

---

155. *Arduino Forum: Open source Project / Hardware* (2007), https://forum.arduino.cc/t/open-source-project-hardware/6861.

156. *Arduino Forum: Arduino vs Genuino* (2015), https://forum.arduino.cc/t/arduino-vs-genuino/347141.

157. Williams, *supra* note 152.

158. *Arduino Forum: Open source Project / Hardware* (2007), https://forum.arduino.cc/t/open-source-project-hardware/6861/75.

159. *Id.*

According to them, Arduino's expansive community had breathed life into the Arduino name and transformed it into a symbol of innovation. As such, users argued that tenets of creation and distribution should be permeated with greater openness, aligned with the ethos that is actually responsible for its evolution.

Along these lines, frustrated users believed that Arduino's technical significance was notable, but ongoing conflicts within the community could lead to their replacement. The community repeatedly underscored that Arduino's true value came from its vast library of functions, user-friendly bootloader,[160] and the dedicated community that has grown around the platform. Users expressed concern that alienating the creators and maintainers of the project could erode the community's support, ultimately impacting Arduino's appeal and success. Further, while users accepted Arduino's motivations for profiting from an open-source project, they felt that attacking others for clones and spinoffs portrayed Arduino negatively.

In an effort to resolve the conflicts and restore unity within the community, the Arduino trademark was ultimately transferred to Arduino Holding, which now acts as the single point of distribution for new products.[161] Simultaneously, the establishment of the non-profit Arduino Foundation was created to oversee community support and the ongoing development of the Arduino IDE.[162] Together, these measures provided much needed clarity and stability for Arduino's future along with a renewed sense of cohesion among its community of users and contributors.

The Arduino trademark tussle also shows the low-road mode in action. To be sure, the "Arduino" trademark was managed by Arduino LLC and the originally Arduino team. But because the trademark was applied to a wide variety of Arduino-compatible software, its use *as a signifier* was largely in the hands of the community of developers creating that software. They were not simply consumers purchasing goods bearing a trademark; they were creators applying a trademark to their own goods to indicate its compatibility.[163]

The split between Arduino LLC and Arduino SRL was thus not just a fight between two producers disputing priority to a mark. Instead, it forced community members to *choose sides* in applying the "Arduino" mark to their own products. There followed a period of rapid ferment, as community members argued over what "Arduino" meant and evolved their labeling practices to communicate different messages. This is the low road: changing meanings

160. Benchoff, *supra* note 151.

161. *Id.*

162. *Id.*

163. In this sense, the trademark functioned more like a certification mark, even though it was registered as a trademark and service mark.

and changing licenses driven by bottom-up decisions by individual actors choosing what licenses and trademarks to apply to their own products.

III. Designing the License, Designing the Community

Legal drafting is a form of design. Like vacuum cleaners and cookie jars, legal instruments are functional artifacts. They exist at all because of what they do, and they are deliberately crafted to do it well. To be sure, legal instruments work through the mechanism of speech; they are texts, and their effects in the world arise because their meaning is translated into action.[164] But there is nothing unique about that. Software is also designed, and it too is inherently textual.

We should expect, then, to see a parallel between the processes that legal drafters follow and the processes that other designers follow. They work creatively within constraints that are not of their choosing. Every project involves a dialogue between their professional expertise and the needs of users. The artifact, once completed, goes out into the world and encounters situations that can be only imperfectly predicted. Product design and legal drafting are close cousins.

What makes open-source communities so interesting, then, is that they are both product designers and legal drafters, and the two are governed through the same institutions and debated in the same fora. Open-source communities are recursive publics with respect to both their technical infrastructure and their legal infrastructure. Their licenses are not just a preexisting armature on which participants hang their contributions. Instead, the licenses themselves are produced by the community through a collective design process that itself conforms in all essential ways to the model of open-source production.

Proprietary licenses have essentially one job: to enact a desired set of legal relations. But open licenses play a central role in articulating the values of the communities that use them. The mutual and reciprocal freedoms enjoyed by members of such a community are ethically laden.[165] The license is a public commitment to those freedoms, and to the values that community members understand those freedoms to embody. Thus, drafting a license is

164. *See* James Grimmelmann, *The Structure and Legal Interpretation of Computer Programs*, 1 J. Cross-Disciplinary Rsch. Comput. L. no. 3, art. 19 (2023).

165. Samir Chopra & Scott D. Dexter, Decoding Liberation: The Promise of Free and Open Source Software (2007); Coleman, *supra* note 8; James Grimmelmann, *The Ethical Visions of Copyright Law*, 77 Fordham L. Rev. 2005 (2009) [hereinafter Grimmelmann, *Ethical Visions*].

a collaborative effort which involves iterative discussion and negotiation, and agreeing upon a license is an act of value articulation.

The ability of licenses to express community values is particularly relevant in the context of copyright law. While copyright law offers licensing options for various technical innovations, it does not itself capture the nuanced values and preferences of communities.[166] Open licenses enable communities to articulate their distinct needs and ideals. Case studies of community-based licenses highlight their role as recursive publics. Investigating open licenses further reveals the significance of the licensing process to the goals of a given community and its innovations. The drafting of licenses involves comprehensive community engagement, fostering self-definition, and governance. Debates, deliberation and contentions become integral tools in the process, enabling communities to collectively shape the license and align it with evolving needs. Further, these articulation processes occur both structurally, within designated platforms such as forums, international conferences, and specific websites, and informally, within the recursivity and interconnectedness of the internet infrastructure. Importantly, the tools (specifically articulation) are activated at moments of ethical, political, or legal crises.

A. Lessons from Open Ecosystems

In the context of open-source software and hardware communities, the process of designing open licenses is intrinsically tied to the shaping of communities and the values they uphold. A compelling illustration of this interplay is visible in the GPLv3 drafting process, where careful planning and extensive community engagement were critical to achieving widespread (though hardly universal) adoption. The FSF attempted to set the stage for an inclusive process by initiating the drafting with the release of a comprehensive "Process Definition" document. It not only outlined the structural components, steps, and anticipated drafts of the revision process but also actively sought input and feedback from the community from the very outset.[167] As the revision process unfolded, the FSF continued to engage with the community through multiples channels, including a STET site, which was instrumental in collecting comments and feedback from community members.[168] The incorporation of clear version control software, along with color-coded comments, proved to be particularly effective in highlighting different opinions, enabling international participation, and sparking high levels of en-

166. Grimmelmann, *Ethical Visions*, *supra* note 165.
167. *GPLv3 Wiki*, *supra* note 74.
168. *Id.*

gagement. The FSF's emphasis on open communication and collaboration ultimately shaped the final version of the GPL, while also cultivating a sense of shared ownership and investment in the resulting license. Although the end product remains controversial, it was the most participatory licensing process to date, and GPLv3 is one of the most widely adopted open software licenses.

Similarly, as the OpenOffice community navigated substantial changes in licensing and governance, the project's technical modularity, coupled with deliberate open licensing decisions enabled volunteers to redefine their community. The new LibreOffice project that they created, under the governance of the Document Foundation, facilitated community engagement through platforms like Internet Relay Chat (IRC) and web forums.[169] On the one hand, IRC provided a real-time, text-based platform where LibreOffice community members could engage in immediate discussions. This allowed for quick problem-solving, brainstorming, and decision-making at a high-stakes moment of reorganization. Developers, contributors, and users who chose to align themselves with the newly created LibreOffice project could rely on IRC to address technical challenges, share ideas, and collaboratively work towards project development. On the other hand, web forums served as a more structured and organized space for in-depth discussions. Users created threads on specific topics, making it easier for the broader community to follow and participate in conversations of interest and importance to them. Web forums provided a format for more detailed and documented exchanges – a space for collaborative ideating and developing best practices for the new organization that extended to questions regarding licensing, project direction, and community values.

Although OpenOffice's Apache License 2.0 is a permissive open license that allows flexibility, it does not require that modifications and derivative works adopt the same license.[170] The dual license, therefore, reflects a specific and situated understanding of the community's needs — one that caters to more flexible branching in the name of community growth and unanticipated uses. The choice to dual-license was not made in isolation; it was a product of meaningful discourse. The community's values of openness and reciprocity were not distant theoretical constructs, but tangible principles actively discussed and deliberated upon — and crucially, which were instantiated in a different way than the FSF community instantiated them. These decisions were bottom-up. They were a result of dialogues, debates, and collective decision-making that aligned with the community's core values.

169. *The Document Foundation Blog - The Home of LibreOffice* (2023), https://blog.documentfoundation.org/.

170. *Apache License, Version 2.0*, *supra* note 110.

MakerBot's approach to community interaction primarily revolved around platforms like blogs, Reddit, and Hackaday, which served as arenas for articulating values, engaging in debates, and challenging existing systems and hierarchies.[171] Blog posts became the medium for sharing thoughts on innovation, design, and the potential of 3D printing technology, articulating community values and aspirations. Reddit served as a space for anonymous open discourse, allowing members of the broader 3D printing community to engage in discussions, express their concerns, and debate the implications of MakerBot's decisions. Importantly, Reddit's format enabled a diverse range of opinions, empowering community members to challenge the status quo and advocate for the principles they believed in. Finally, Hackaday was the innovation hub. Here, community members showcased their creative designs, modifications, and hacks that tangibly demonstrated their commitments to their core values of openness and reciprocity.

However, the MakerBot community experienced a notable exclusion from the license drafting process. This exclusion both diminished the community's influence on the project's legal framework, and disrupted the alignment of community values and norms essential for self-regulation. Ultimately, the absence of community participation in shaping the license obstructed the community's ability to participate and steer the project's direction. As a result, many members no longer saw it as an open and cohesive self-regulating community. Despite Makerbot's top-down decision-making, the 3D-printing community managed to preserve its recursivity by employing online platforms and community-driven strategies that had served it prior to Makerbot's success. In other words, the goals, norms, and values of creation and sharing were already outlined and documented by RepRap. They were entrenched.[172]

In the context of Arduino, the site for contention revolved around trademark inconsistencies that significantly impacted project governance and support. This period was marked by the proliferation of off-brand duplicates of the Arduino board, which introduced considerable confusion within the Arduino community. As a consequence, opinions differed about which Arduino organization to trust following the division within the community. The mistrust in leadership, compounded by uncertainties in the Arduino board's manufacturing processes further eroded opportunities for the community to engage in productive collaborative efforts. However, as the different Arduino camps gradually worked towards a resolution and centralized their organizational structure, the community's shared objectives and defined norms began to restore collaboration. This process of resolution and

171. Benchoff, *supra* note 135.
172. *RepRap Policy* (2018), https://reprap.org/wiki/Policy.

centralization allowed the community to rebuild trust and regain a sense of cohesion. In doing so, the Arduino community revived its capacity for co-creation and rekindled its commitment to the principles initially established, thus strengthening the recursive and self-regulatory aspects within the community.

Arduino's case exemplifies the significance an adaptable community framework, akin to the architectural modularity in buildings, where distinct components can be worked on independently or collaboratively, while still being integrated into an overarching project. This capacity for recursion and self-regulation is crucial in maintaining the community's cohesion, ensuring that it can navigate challenges and disputes while continuing to pursue its shared goals and values. Sometimes these processes are community driven and bottom-up. Other times they are top-down, a function of a project's organized governing body that centers the needs of a changing community. Together, they underscore how the structure and dynamics of open source ecosystems as reflected through open licensing and community-driven governance, significantly influence the direction and ethos of open technical projects.

B. Community Building and Governance

Across open collaborative communities, governance models have an essential role in shaping the behaviors and interactions of its stakeholders. Drawing on Ostrom's governance framework which underscores informal norms, rules, and principles, participatory license design processes can become a functional tool for technical communities to adopt a value-centered approach to governing their ecosystem. Take, for example, the principle of reciprocity embedded within the GPL. This clause mandates that any software distributed under the GPL must be reciprocated under the same licensing terms. Here, the importance of sharing and equitable distribution of collaborative software innovations is fundamental. It ensures that both long-standing and new community members enjoy the fruits of collective labor.[173]

In the realm of open software development, self-regulation of the commons is facilitated through mechanisms of agreement and norm establishment. When the knowledge commons faces threats or disruptions, open software communities may work towards reinstating the norms and rules that govern the commons, or replace them with new ones. Existing infrastructure built under previous regimes serves as a foundation for navigating and adapting to new eras of development, as seen in both open software case studies.

173. *The GNU General Public License v3.0*, *supra* note 18.

In contrast, the open hardware domain has been characterized by lesser political engagement and a relatively weaker capacity to build and negotiate social infrastructure around its commons.[174] These disparities can be attributed to the non-zero marginal cost of copying hardware, which undermines the perception of hardware as a freely and easily shareable resource - a phenomenon we observe within the open software community. The robustness of intellectual property management tools combined with shared values and norms play a crucial role in a community's ability to handle ripples and shocks. Consequently, when crises arise within the open hardware community, their recovery and management processes becomes considerably more challenging.

In both domains, the license, as a recursive infrastructure, serves as a mechanism that sustains the community's position as a recursive public. The process of writing the license forces the community to critically examine its own boundaries and scope. The resulting license document serves as an anointment for new members as well requiring their commitment and adherence to the community's guiding ethos. Similarly, the license protects the norms of peer production, characterized by a reliance on open platforms and tools, volunteerism, self-organizing, and self-governance.[175] As the mode of production where volunteers employ shared resources to create knowledge, peer production is activated by licensing regimes that support sharing. Collective intelligence enables the community to tap into the distributed knowledge and expertise of a community to create innovations that would be difficult or impossible to produce using traditional hierarchical models of organization.[176] Creative Commons licenses, for example, helped bring "reasonableness" into copyright and enacted the ability to create and freely collaborate, establishing new forms of social organization.[177]

Licenses are powerful tools for community building, going beyond their legal implications. Licenses serve as rallying points, bringing together diverse stakeholders to collaboratively negotiate terms and shape a project's trajectory. This collective effort cements the community's persistence even when a specific project or product fades away, as the shared values and interests endure. In the realm of software, open licenses like the GPL have played a significant role in supporting vibrant communities such as LibreOffice. However, the situation is more intricate in the domain of open hardware. MakerBot's transition away from open licensing led to concerns about community

174. Katz, *supra* note 122.

175. Benkler & Nissenbaum, *supra* note 34.

176. Yochai Benkler, Aaron Shaw & Benjamin Mako Hill, *Peer production: A form of collective intelligence*, 2015 Handbook collective intelligence 175.

177. Lawrence Lessig, *supra* note 4.

engagement and the project's direction. The absence of a widely-accepted open hardware license complicates the process of community convening and negotiation, where striking a balance between openness and commercial interests is an ever-present challenge.

Open licensing, therefore, serves as an essential form of commons management or self-governance infrastructure. The processes of license drafting and application are the crucial mechanisms through which communities design and regulate the use of their creative intellectual output. Collaborative endeavors within this realm involve iterative discussions and negotiations between community members to define the terms of sharing and utilization. The license becomes a design-based infrastructure that carries not only legal significance, but also encompasses intellectual resources, and shared values and goals. The deliberative process enacts the community's agency, and self-regulating intellectual property rights ensure the preservation of knowledge and objectives. Ultimately, the license acts as a visible symbol that represents the community's commitment to an environment of cooperation, innovation, and the advancement of intellectual resources. It is through the careful crafting of licenses that communities empower themselves to shape the trajectory of their intellectual infrastructures, enacting an ecosystem that thrives on collective ideals.

Open licensing infrastructure is malleable and mirrors the flexible nature of the technical projects. These licenses embrace adaptability, enabling projects to evolve and respond to changing needs, circumstances, and innovations. The malleable nature of open licensing not only empowers creators and contributors to define and articulate how their intellectual property is governed, but also reinforces the spirit of collaboration and continuous improvement. Whereas buildings "learn" through their symbiotic relationship with users, technical projects "learn" through the collective knowledge and expertise of the community which is enacted and enshrined in their designed licenses.

Conclusion

Open licenses are infrastructures that build communities as they are built by communities. This essay presents a roadmap for understanding how communities that design also design themselves by aligning social values and legal tools. Our case studies describe how communities insightfully self-regulate their intellectual property. We highlight the recursive technical infrastructure of open licenses, and the interplay between community needs, and the tools they employ to articulate and center their objectives and goals.

Buildings exhibit recursion when they adapt and evolve over time, responding to the changing needs of their inhabitants, inhabitants who then maintain and serve the interests of the building. Technical infrastructure, such as software and hardware projects, also undergo recursive transformations as communities explore new uses and functionalities. Licenses, as an essential form of recursive infrastructure embody and formalize how a community defines itself.

Lawrence Lessig's *Code* is today best remembered for its argument that the metaphorical "architecture" of the Internet is both a substitute for law and can be regulated by law. But it is also a subtle and careful meditation on the nature of architecture itself in digital spaces, and *Code*'s footnotes are filled with citations to architectural theory. This essay is offered in the same spirit. Understanding how physical infrastructure adapts to new challenges helps us understand how legal infrastructure does the same.