JIRRS
Journal International Review of Research Studies

Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

OPEN SOURCE VS. PROPRIETARY SOFTWARE

Anthony Santos Batista
University AlfaUnipac, Brazil
Corresponding author: contato.anthonysb@gmail.com

Abstract

The tension between open source and proprietary software constitutes one of the defining structural conflicts of the contemporary digital economy — a conflict that is simultaneously technical, economic, political, and philosophical. This article presents a comprehensive comparative and analytical mapping of both paradigms across their technical, economic, organizational, governance, social, and geopolitical dimensions, drawing on peer-reviewed scholarship from information systems research, political economy, organizational theory, science and technology studies, and legal analysis. The analysis traces the historical emergence of proprietary software as a commercial category following IBM's 1969 unbundling decision; the countervailing institutionalization of the free software and open source movements from 1983 onward; and the subsequent decades of competitive and collaborative interaction that have produced the hybrid landscape of the present. Across the technical domains examined — operating systems, databases, programming languages, cloud infrastructure, development tooling, artificial intelligence, and cryptographic security — the article identifies evidence consistent with a structural asymmetry: open source has achieved categorical dominance at the infrastructure layer of the digital economy, while proprietary models maintain dominance at the interface and service layers where economic and political power is concentrated. Key tensions analyzed include the free rider problem and chronic underinvestment in open source commons; the strategic commoditization logic by which large technology corporations selectively open-source components to erode competitors' proprietary advantages while retaining proprietary control at value-capturing layers; the governance risks of maintainer burnout, corporate capture, and succession failure; the emerging conflicts between regulatory frameworks — including the EU AI Act, the Cyber Resilience Act, and data protection legislation — and the open source ecosystem; and the geopolitical dimensions of digital sovereignty in which the open/proprietary choice has become a matter of national strategic policy. Six paradigmatic case studies — Linux versus Windows Server, Android's open core architecture, HashiCorp's relicensing of Terraform, the Redis and Elasticsearch licensing conflicts with Amazon Web

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Services, Meta's LLaMA strategy, and Kubernetes' multi-stakeholder governance — are analyzed to ground the theoretical framework in empirically documented outcomes. The article concludes that the productive question is not which model is superior in the abstract, but rather who benefits from each model, under what institutional conditions, and at which layer of the technology stack — a distributional and contextual analysis that reveals the open source versus proprietary debate to be, at its deepest level, a contest over who holds the power to define how the digital world functions.

Keywords: open source software; proprietary software; digital sovereignty; software governance; platform economics; vendor lock-in; open core business model; artificial intelligence regulation; EU AI Act; software supply chain security; technology policy; political economy of technology; free rider problem; commons governance; maintainer sustainability; cloud computing; open-weight AI models; enshittification; Global South digital infrastructure; software licensing

1. Introduction

The history of software development has been shaped by a persistent and consequential tension between two fundamentally different paradigms: open source software (OSS), grounded in principles of transparency, collective ownership, and freedom of use; and proprietary software, characterized by restricted access, centralized control, and commercial licensing. This dichotomy is not merely technical — it is ideological, economic, and increasingly geopolitical. As digital infrastructure becomes ever more central to the functioning of economies, governments, and societies, the choice between open and closed software models carries profound implications for who controls technology, who benefits from it, and who is excluded (Bechara & Lechner, 2024; Biström et al., 2024).

A provocative starting point for this debate lies in a seemingly paradoxical observation: the overwhelming majority of the digital infrastructure that runs the modern world is built on open source foundations. Linux, the open source operating system developed by Linus Torvalds in 1991, currently powers approximately 53% of all servers globally and runs on over 96% of the world's top one million web servers. All 500 of the world's fastest supercomputers operate on Linux distributions, a dominance that has persisted without interruption since 2017. Cloud infrastructure — the backbone of the digital economy — relies on Linux for more than 90% of its public workloads. And yet, despite this pervasive adoption, the debate between open source and proprietary models remains vigorously alive, continuously reinvigorated by shifts in technology, market structure, and political economy. The question, then, is not

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

simply which model has "won," but rather what the persistence of this conflict reveals about the structural dynamics of technological power (XtendedView, 2025).

The roots of this conflict are found in the early 1980s, when the increasing commercialization of software prompted Richard Stallman to establish the GNU Project in 1983 and the Free Software Foundation in 1985, articulating a moral and political philosophy of software freedom that opposed the emergent proprietary model (Tolu, 2018). A decade later, Eric S. Raymond's seminal essay "The Cathedral and the Bazaar" (1997) reframed the argument in engineering and economic terms, helping to bring the open source movement into mainstream commercial acceptance and coining the terminological shift from "free software" to "open source" that occurred in 1998 (Tolu, 2018). Since then, open source development has transitioned from a countercultural movement into a mainstream industrial practice. Empirical evidence suggests that the strategic importance of open source to enterprise IT infrastructure has grown significantly: the growing acceptance of open source software as a viable part of enterprise information technology infrastructure is an interesting contemporary phenomenon that continues to draw the attention of researchers and practitioners (Esseola et al., 2015). Research from the mid-2010s onward documents how firms with stronger IT capabilities adopt open source more broadly, while the heterogeneity of software infrastructure — combining both proprietary and open source components — is in fact the most common organizational configuration (Wen & Choudhury, 2019).

The debate, however, has not been resolved by these structural shifts. If anything, it has intensified along new frontiers. Three contemporary developments have dramatically re-energized the open source versus proprietary conflict. First, the rapid expansion of cloud computing has introduced a new form of platform dependency in which major providers — Amazon Web Services, Microsoft Azure, and Google Cloud — leverage open source software at their infrastructure layer while maintaining proprietary control over services, pricing, and data access, raising novel questions about vendor lock-in and technological sovereignty (Bechara & Lechner, 2024). Second, the emergence of artificial intelligence (AI) as a core technological domain has created a new theater of contestation: since 2022, the number of publicly released AI models has more than doubled, and more of them now come with open weights, giving developers far more transparency and control, while open models are quickly closing the gap with proprietary systems (European Commission, 2025). Third, concerns over digital sovereignty — the ability of states and organizations to retain autonomous control over their digital infrastructure — have placed the open source versus proprietary debate at the center of national and supranational policy agendas, particularly in Europe, where the European Commission has formalized an Open Source Software Strategy (2020–2023) and where dependency on foreign-

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

controlled proprietary platforms has emerged as a strategic vulnerability (Biström et al., 2024; Bechara & Lechner, 2024).

The analysis proceeds through several interrelated sections: following this introduction, Section 2 provides a methodological clarification of the nature and scope of the analysis, its operational definitions, and its principal limitations; Section 3 develops the theoretical framework and epistemological positioning, tracing the philosophical, economic, and geopolitical foundations of both paradigms; Section 4 presents a historical overview of the co-evolution of open source and proprietary software from IBM's 1969 unbundling decision through the current AI era; Section 5 maps the technical landscape across seven domains — operating systems, databases, programming languages, cloud infrastructure, development tooling, artificial intelligence, and security — in which the open source versus proprietary contest is actively playing out; Section 6 analyzes the economic dimensions of both models, including business model archetypes, total cost of ownership, and market consolidation dynamics; Section 7 examines organizational and governance dimensions, including open source project governance, corporate adoption strategies, and the governance risks of proprietary software; Section 8 addresses the social, cultural, and political impact of both paradigms across education, digital sovereignty, diversity, and environmental sustainability; Section 9 presents six paradigmatic case studies; Section 10 surveys the debate across specific sectors including healthcare, education, government, finance, and artificial intelligence; Section 11 identifies emerging trends and future directions; and Section 12 presents the conclusions.

2. Methodological Clarification

This article is conceived as a comparative and analytical inquiry rather than an advocacy document. Its primary objective is to map, systematically and critically, the key dimensions along which open source software (OSS) and proprietary software differ, compete, and increasingly converge — without presupposing the superiority of either model. The analysis draws on established frameworks from information systems research, organizational theory, political economy, and science and technology studies in order to treat both paradigms as complex social and economic phenomena rather than mere technical alternatives (Fitzgerald, 2006; West & Gallagher, 2006).

To provide systematic structure beyond narrative description, the comparative analysis follows a Multi-Criteria Decision Analysis (MCDA) logic (Belton & Stewart,

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

2002), evaluating both paradigms across five criteria: (1) Technical Performance, (2) Security and Resilience, (3) Economic Viability and Total Cost of Ownership, (4) Governance and Transparency, and (5) Sovereignty and Strategic Autonomy. Each criterion is assessed at three levels — organizational, sectoral, and policy — reflecting the empirical finding that relative advantages differ systematically depending on the unit of analysis (West & Gallagher, 2006).

Case and domain selection followed three explicit criteria. Inclusion: a case is included only if documented in peer-reviewed literature or a verifiable institutional source, assessable across at least three of the five criteria, and representative of a broader governance pattern rather than an isolated anomaly. Exclusion: cases documented only in vendor-produced sources or lacking independently verifiable empirical data are referenced descriptively only. Domain coverage: the selection across Sections 5–10 is designed to cover each of the seven technical domains identified in the scope statement, with at least one longitudinal case study per major category.

The analytical pipeline proceeds in five stages. Stage 1 — literature identification: systematic searches of ACM Digital Library, IEEE Xplore, Web of Science, and Scopus using the terms 'open source software,' 'proprietary software,' 'software governance,' 'digital sovereignty,' and 'open source AI,' combined with domain-specific terms for each technical area in Section 5, covering literature from 2000 to early 2026. Stage 2 — source classification: each source is classified as primary empirical, secondary analytical, or tertiary descriptive; claims are qualified accordingly. Stage 3 — criterion mapping: each case is mapped against the five MCDA criteria, recording direction of advantage, strength of evidence, and boundary conditions. Stage 4 — cross-domain synthesis: patterns are identified where at least three independent domain analyses and two source tiers converge; disagreements are flagged as context-dependent rather than resolved. Stage 5 — boundary conditions: each analytical section states explicitly the conditions under which its findings hold, operationalising the external validity limitation described above.

The scope of the analysis is deliberately delimited. The article focuses on three principal domains: (1) general-purpose software, including operating systems, infrastructure tooling, databases, and development frameworks; (2) artificial intelligence models and platforms, with particular attention to the emerging distinction between open-weight and closed proprietary AI systems; and (3) digital platforms and cloud services, where the interaction between open source foundations and proprietary service layers generates a new set of structural tensions. Adjacent debates — such as open hardware, open data, and open access in academic publishing — are acknowledged where directly relevant but are not the primary

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

subject of analysis. The empirical base mobilized throughout the article consists of peer-reviewed academic literature, industry research reports from established market research institutions, and documented case studies of major technology organizations. The literature reviewed is not uniformly convergent: on several key questions — including whether open source adoption improves organizational security outcomes, whether open innovation theory adequately explains corporate open source strategy, and whether digital sovereignty framings serve democratic or authoritarian ends equally — the evidence base reflects genuine scholarly disagreement that is acknowledged throughout the analysis rather than resolved by fiat. Where quantitative data are cited, their methodological provenance is noted (Ven & Verelst, 2008). In the interest of transparency and replicability, all primary data sources mobilized in this article are publicly available. The quantitative adoption figures cited in Section 5 derive from the following publicly accessible datasets and reports: TOP500 supercomputer statistics (top500.org, November 2024 edition); StatCounter Global Stats desktop OS market share (gs.statcounter.com, February 2025); CNCF Annual Survey 2024 (cncf.io/reports); XtendedView Linux Statistics 2025 (xtendedview.com/linux-statistics); and the Synopsys OSSRA Report 2024 (synopsis.com/software-integrity). The security incident metrics cited in Section 5.7 derive from CVE records maintained by NIST (nvd.nist.gov) and from the primary empirical studies cited in each case (Durumeric et al., 2014; Zeber et al., 2015; Ghanbari et al., 2024; Pirozzi et al., 2025). No proprietary datasets were used. A structured mapping file documenting source classifications, criterion assignments, and boundary condition statements for all cases analyzed is available as Supplementary Material S1 at [https://github.com/anthony-santos-batista/ai-benchmark-study]. As this article is a comparative analytical study rather than a computational experiment, no software scripts or model configurations were produced; the analytical pipeline is described in full in Section 2.6.

Definitional precision is essential in this field, as terminological ambiguity has repeatedly generated confusion in both academic and policy debates (Bonaccorsi & Rossi, 2003). Six terms in particular are used inconsistently in the literature and require explicit operational definitions for the purposes of this article.

Security is used here exclusively to refer to the technical and organizational properties that reduce the likelihood and impact of unauthorized access, data breach, supply chain compromise, or malicious code execution in a software system. It does not encompass privacy (the protection of personal data from legitimate but unwanted use), nor confidentiality of model weights (a distinct concern specific to AI systems addressed in Section 5.6). Where the literature uses 'security' in a broader sense, this article notes the distinction explicitly.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Transparency refers, depending on context, to one of three distinct phenomena that must not be conflated: (a) *code transparency* — the public availability of source code under an inspectable license; (b) *process transparency* — the public documentation of development decisions, governance structures, and contribution histories; and (c) *data transparency* — the disclosure of training datasets, data pipelines, and evaluation methodologies, relevant primarily to AI systems. Each sub-dimension is specified when the term is used in Sections 5 through 11.

Control is used in this article to refer to the capacity of a defined actor to make binding decisions about a software system's development, deployment, or termination. The actor in question varies by context and is always specified: *developer control* refers to the ability to modify source code; *user control* refers to the ability to deploy, audit, or migrate away from a system without vendor permission; and *regulatory control* refers to the authority of public institutions to audit, mandate, or restrict a system's operation. Claims about 'who controls' a technology are uninterpretable without specifying which of these dimensions is at stake.

Open source software is understood in a sense that extends beyond the mere technical condition of code availability. At its legal minimum, OSS refers to software distributed under a license satisfying the Open Source Definition maintained by the Open Source Initiative, which requires, among other conditions, free redistribution, availability of source code, allowance of derived works, and non-discrimination against persons, groups, or fields of endeavor (Perens, 1999). However, this article adopts a broader conception, consistent with O'Mahony's (2007) analysis, which recognizes that the concept of open source initially referred to software projects managed by grassroots communities in public forums, and that since 1998 the concept has been adapted and extended to new settings, with more private parties contributing to OSS communities and more hybrid governance models emerging (O'Mahony, 2007). Accordingly, open source is treated here as a governance regime as much as a licensing category — one that encompasses dimensions of transparency, community participation, and the right to fork or modify the codebase, regardless of whether the project is maintained by a volunteer collective, a non-profit foundation, or a for-profit corporation (Fitzgerald, 2006; Shah, 2006).

Proprietary software is defined as software in which the source code is withheld from users, and in which the rights to use, modify, distribute, and inspect the software are controlled by a legal rights-holder, typically under a commercial license (Bonaccorsi & Rossi, 2003). This definition encompasses a historically significant range of product forms: from traditional packaged software sold as perpetual licenses, to subscription-based enterprise software, to modern Software as a Service (SaaS) platforms in which the software is never distributed to the user but is instead accessed

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

remotely via a network interface (Bechara & Lechner, 2024). The proprietary SaaS model is particularly consequential for the contemporary debate because it relocates the locus of control from source code access to service architecture, data portability, and contractual terms — dimensions that are not fully captured by the classical licensing framework (Biström et al., 2024; Bechara & Lechner, 2024). The diversity among available cloud SaaS services has led to proprietary architectures being used by cloud vendors, increasing the risk of vendor lock-in for customers who encounter significant challenges to migrating and interconnecting services (Petcu et al., 2013).

Gray zones in the typology deserve particular attention, as they represent some of the most commercially significant software categories in the current ecosystem. Three models occupy an ambiguous position between the open and proprietary poles. First, *source-available* software makes source code readable but restricts redistribution, modification, or commercial use, typically through licenses such as the Business Source License or the Server Side Public License — neither of which qualifies as open source under the Open Source Initiative definition (Riehle, 2009). Second, the *freemium* model provides a base version of software at no cost, with the free tier functioning primarily as a customer acquisition mechanism rather than a reflection of genuine openness, while the core product may remain fully proprietary (West & Gallagher, 2006). Third, and most consequential for the contemporary market, the *open core* model — theorized by Riehle (2009) — involves a firm releasing a feature-limited version of its software under an OSI-approved license while simultaneously commercializing an enterprise-grade version under a proprietary license. The financial models supporting open source development have diversified substantially, with early projects relying on volunteer contributions and non-profit foundations, while more recent commercial adoption has produced profit-oriented models including professional services, dual licensing, open core structures, and cloud service offerings (Fitzgerald, 2006; Riehle, 2009). These hybrid configurations demonstrate that the open source versus proprietary distinction is not binary but scalar, and that many of the most influential technology firms of the past decade occupy a deliberately ambiguous position between the two poles (Shah, 2006; West & Gallagher, 2006).

Six methodological limitations merit explicit acknowledgment before the substantive analysis proceeds.

First, there is a structural risk of false dichotomy inherent in any comparative framing of open source and proprietary software. Empirical research consistently shows that most organizations do not adopt a single model exclusively, but rather operate heterogeneous IT environments in which open source and proprietary components coexist and are mutually dependent (Wen & Choudhury, 2019). Treating open and

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

proprietary as mutually exclusive alternatives would therefore misrepresent the organizational reality of software adoption. Throughout this article, hybrid configurations are acknowledged and, where evidence permits, analyzed as a distinct category (West & Gallagher, 2006).

Second, the pace of change in the software ecosystem imposes severe constraints on the durability of any empirical claim. Licensing terms, market positions, governance structures, and industry norms shift rapidly and sometimes discontinuously — as illustrated by high-profile relicensing decisions by firms such as HashiCorp, Redis, and Elasticsearch in the early 2020s, each of which migrated from OSI-approved licenses to more restrictive source-available terms in response to competitive pressures from cloud providers (Bechara & Lechner, 2024). Findings that are empirically grounded at the time of writing may therefore require revision as the ecosystem evolves, and the article makes no claim to offer a static or permanently valid settlement of the debate.

Third, survivorship bias represents a significant methodological hazard in the academic literature on technology adoption. Studies of successful open source projects — large, well-maintained, widely adopted codebases — are far more common than studies of the much larger population of abandoned or failed OSS initiatives, and similarly, accounts of successful proprietary products dominate over analyses of commercial software that failed to achieve adoption (Ven & Verelst, 2008). This asymmetry inflates estimates of the average quality, sustainability, and economic viability of both models, and should be borne in mind when interpreting the comparative evidence presented in subsequent sections.

Fourth, case and model selection bias represents a structural limitation of the comparative analysis. The six paradigmatic case studies examined in Section 9 — Linux, Android, HashiCorp/Terraform, Redis/Elasticsearch, Meta/LLaMA, and Kubernetes — were selected because they are extensively documented in peer-reviewed literature and represent distinct governance archetypes. They are not a random or representative sample of all open source or proprietary software deployments. Projects that failed quietly, were never widely adopted, or operate in non-English-language ecosystems are systematically underrepresented in the available literature and therefore in this analysis. Readers should not generalize from these cases to the full population of software projects without accounting for this selection effect.

Fifth, domain specificity constrains the transferability of the findings. The analysis focuses on general-purpose infrastructure software, AI foundation models, and cloud platforms — domains characterized by large contributor communities, significant

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

corporate investment, and high public visibility. The conclusions drawn may not apply with equal force to embedded systems, scientific software, domain-specific enterprise applications, or software developed in low-resource institutional contexts, where the governance dynamics, security threat models, and economic structures differ substantially from those documented here.

Sixth, the pace of technological change imposes a temporal boundary on all empirical claims in this article. The open source versus proprietary landscape in artificial intelligence in particular has shifted substantially within single calendar years: the release of Llama 3.1 in July 2024, the enactment of the EU AI Act in August 2024, and the publication of the Open Source AI Definition in October 2024 each altered the terms of the debate in ways that could not have been anticipated twelve months prior. Findings that are empirically grounded at the time of writing should be treated as a time-stamped snapshot rather than durable conclusions, and the article makes no claim that the structural patterns it identifies will persist unchanged beyond the near term.

The external validity of this study's conclusions is bounded by its analytical scope. The comparative patterns documented — open source dominance at the infrastructure layer, proprietary dominance at the interface layer, hybrid configurations as the organizational norm — are well-supported within the domains examined. They should not be extrapolated as universal laws of software development, nor applied without adjustment to jurisdictions, sectors, or organizational contexts not represented in the evidence base reviewed.

3. Theoretical Framework and Epistemological Positioning

3.1 Epistemological Positioning

The conflict between open source and proprietary software is not reducible to a technical preference or a market outcome. Epistemologically, this article is positioned as an analytical-comparative study rather than an exploratory or purely descriptive inquiry: it assumes that open source and proprietary software are sufficiently documented phenomena to permit systematic comparison across defined criteria, while acknowledging that causal claims about governance outcomes remain tentative given the observational nature of the available evidence. It is, simultaneously, a technical phenomenon, an economic competition, a political struggle, and a philosophical dispute about the nature of knowledge and the ethics of its control. To analyze it as merely one of these dimensions at the exclusion of others would produce

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

a distorted account. The theoretical framework adopted in this article therefore treats software development models as inherently multidimensional objects of inquiry — one that demands tools from information systems research, political economy, philosophy of technology, and international relations theory in equal measure (West & Gallagher, 2006; Bonaccorsi & Rossi, 2003).

A foundational claim of this framework is that technological choices are never neutral. As Langdon Winner argued in his foundational essay "Do Artifacts Have Politics?", the machines, structures, and systems of modern material culture can be accurately judged not only for their contributions to efficiency and productivity, but also for the ways in which they embody specific forms of power and authority (Winner, 1980). Applied to software, this insight implies that the decision to release code as open source or to retain it as proprietary is never a purely technical act — it is always, simultaneously, a choice about who may inspect, modify, reproduce, and build upon a piece of technology, and therefore about how power is distributed among users, developers, vendors, and states (Winner, 1980; Bowker & Star, 1999). The concept of infrastructure is equally central to this epistemological positioning. Drawing on the work of Star and Ruhleder (1996) and Bowker and Star (1999), infrastructure is understood here not as a fixed material substrate but as a relational concept: something becomes infrastructure when it achieves a particular function for a community of users, rendering itself invisible through smooth and habitual use while embedding specific norms, values, and dependencies in the process (Bowker & Star, 1999). Software — particularly operating systems, databases, and AI frameworks — qualifies as critical infrastructure in precisely this sense: it is deeply embedded, largely invisible to end users, and extraordinarily difficult to replace once organizational dependencies have formed around it (Star & Ruhleder, 1996). This infrastructural character enormously raises the stakes of the open source versus proprietary choice, because the selection of a software paradigm at the infrastructure layer shapes and constrains every subsequent layer of technological and organizational practice built upon it.

3.2 Philosophical Foundations of Open Source

The philosophical foundations of the open source movement were laid in the early 1980s by Richard Stallman, then a researcher at the MIT Artificial Intelligence Laboratory. Confronted with the increasing commercialization and legal restriction of software that had previously circulated freely among researchers and developers, Stallman launched the GNU Project in 1983 with the explicit goal of developing a complete free operating system. In 1985, he published the GNU Manifesto in *Dr. Dobb's Journal*, which articulated the ethical case for software freedom. The GNU Manifesto explicitly invokes the Golden Rule as the ethical basis for the imperative to

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

share: Stallman argued that if he liked a program he was required by this principle to share it with others, and that software sellers who sought to divide users from one another — making each agree not to share with others — were engaged in a socially destructive act (Stallman, 1985). The moral architecture of Stallman's argument is deontological rather than consequentialist: it grounds the obligation to share code not in a calculation of aggregate welfare but in a principle of ethical consistency and solidarity among members of a technological community (Stallman, 1985; Tolu, 2018).

A conceptually significant distinction runs through the history of the movement: the difference between *free software* and *open source*. Stallman's Free Software Foundation defines free software through four essential freedoms — to run the program, to study and modify the source code, to distribute copies, and to distribute modified versions — and frames this as an irreducibly ethical position. As the FSF's own materials make clear, it is a mistake to associate the GNU Project with the term "open source," which was coined in 1998 by people who deliberately intended to distance themselves from the politically and philosophically charged language of the free software movement (Free Software Foundation, 1985/2024). The term "open source" was introduced precisely to make the practical benefits of collaborative development palatable to corporate audiences without committing to Stallman's ethical premises (Tolu, 2018). This distinction — between free software as a social movement grounded in ethics, and open source as a development methodology grounded in pragmatics — remains consequential: it explains why ostensibly "open source" projects can be governed by corporations in ways that the free software movement regards as ethically inadequate, even when they technically satisfy OSI licensing criteria (Fitzgerald, 2006; Shah, 2006).

The second foundational philosophical text of the open source tradition is Eric S. Raymond's essay "The Cathedral and the Bazaar," first presented in 1997 and published in book form in 1999. Drawing on his experience managing the *fetchmail* email utility and his observation of the Linux kernel development process, Raymond proposed a systematic contrast between two models of software development. The cathedral model — characteristic of most commercial and institutional software — involves small teams of experts developing software in isolation before releasing completed versions to the public. The bazaar model — characteristic of Linux and the open source tradition — involves releasing early and frequently to a large, distributed community of contributors, treating users as co-developers, and relying on the law of large numbers to surface and resolve bugs at a pace impossible for any closed team (Raymond, 1999). Raymond's central epistemological claim — "given enough eyeballs, all bugs are shallow" — is not merely a software engineering proposition: it is a claim about the epistemic superiority of distributed, transparent peer review over

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

centralized expert authority, and as such carries implications well beyond software (Raymond, 1999). The analogy with the public domain and the knowledge commons is explicit in Raymond's framework: software, he argues, is most effectively improved when it is treated as a collective intellectual resource rather than a proprietary asset, in a manner structurally analogous to how open scientific literature enables cumulative knowledge production (Raymond, 1999; Bonaccorsi & Rossi, 2003).

3.3 Philosophical Foundations of Proprietary Software

The philosophical foundations of the proprietary software model rest on three mutually reinforcing pillars: intellectual property theory, economic incentive logic, and a product philosophy centered on quality control, accountability, and user experience.

The intellectual property justification for proprietary software derives from a broader theory of rights in intangible goods, which holds that creators who invest resources in producing original works require legally enforced exclusivity to recoup those investments and generate returns sufficient to justify further creation. Intellectual property rights are traditionally justified as a mechanism for generating incentives to innovate, with the ultimate goal being the public good through the promotion of progress for the benefit of society (Huang et al., 2013). In the context of software, copyright law — which attaches automatically upon the creation of original code — and trade secret law provide the principal instruments of this exclusivity, preventing unauthorized copying, reverse engineering, or redistribution (Gliściński, 2025). The rights-holder then licenses access to the software's functionality on contractual terms, generating revenue streams that are intended to fund continued development. This market-based model of software production was described by Baldwin and von Hippel as the "producer model," in which an individual entity uses intellectual property rights to control the entire production and distribution process of intangible goods (Gliściński, 2025). It is worth noting that this rights-based justification is contested in the literature: critics argue that software copyright terms far exceed what is necessary to incentivize development, and that the expansion of intellectual property protections over the past four decades has in several documented cases impeded rather than accelerated cumulative innovation (Benkler, 2006; Stiglitz, 2008).

The economic argument for proprietary software goes beyond legal rights to make a claim about innovation dynamics. Once an innovation has been created, its non-rival character suggests that its benefits would be maximized if use were free to all at marginal cost; however, such availability would severely damage the incentive for further investment in innovation (UNIDO, 2003). The proprietary model therefore accepts a short-run efficiency cost — restricting access through pricing and licensing — in exchange for the long-run dynamic benefit of sustained investment in research

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

and development. This trade-off is the classical justification for intellectual property systems more broadly, and it applies with particular force to software, where development costs are high, marginal costs of distribution are near zero, and the potential for free riding without legal protection is substantial (Stiglitz, 2008). It must be noted, however, that the empirical evidence supporting this justification is considerably more mixed than the theoretical argument suggests: the relationship between stronger intellectual property protection and increased innovation is conditional on the level of development and organizational context, and excessive protection can in fact impede cumulative innovation by restricting access to prior knowledge that later innovators need as a foundation (Huang et al., 2013; UNIDO, 2003).

The third pillar — the product philosophy — offers a pragmatic rather than rights-based defense of proprietary development. Centralized control over a codebase enables consistent quality assurance, defined accountability structures, and coherent user experience design in ways that decentralized community development may struggle to replicate (Bonaccorsi & Rossi, 2003). The proprietary vendor, by controlling the full development lifecycle, can make architectural decisions with a long-term vision, allocate dedicated quality engineering resources, and offer legally binding service level agreements to enterprise customers — commitments that volunteer-driven open source projects, however technically excellent, structurally cannot make (West & Gallagher, 2006). This constitutes what may be described as an implicit social contract embedded in the proprietary model: users pay not merely for functional software, but for reliability, support, and the transfer of risk from the user to the vendor (Fitzgerald, 2006).

3.4 Relevant Economic Theories

Several theoretical frameworks from economics bear directly on the open source versus proprietary debate and provide analytical tools for understanding why both models persist, how they interact, and why neither has entirely displaced the other.

The theory of public goods is foundational. Software, unlike most physical goods, exhibits the two classical properties of public goods: non-rivalry, meaning that one person's use of a piece of software does not diminish its availability to others, and non-excludability, which is structurally present in software once source code is made public and which proprietary models attempt to overcome through legal instruments rather than physical scarcity (Bonaccorsi & Rossi, 2003). The public goods character of software generates a classic collective action problem: because the benefits of developing a piece of software can in principle be enjoyed by all, individual actors have rational incentives to free ride on others' development efforts rather than

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

contribute themselves. This is the theoretical basis for the "free rider problem" in open source, which Lerner and Tirole (2002) identified as one of the central economic puzzles of open source production: why do rational, self-interested agents contribute to a resource from which all may benefit without contributing? Their answer invokes a combination of reputational incentives, career signaling, and intrinsic motivation — all of which represent departures from the simplest model of rational self-interest (Lerner & Tirole, 2002; Shah, 2006).

Elinor Ostrom's theory of the commons, developed through decades of empirical study of natural resource governance and awarded the Nobel Prize in Economic Sciences in 2009, offers a powerful theoretical framework for understanding how open source communities manage shared codebases. Ostrom demonstrated, against the prevailing assumption of the "tragedy of the commons," that communities are frequently capable of developing self-governing institutional arrangements that sustain shared resources without either privatization or state regulation — provided that certain design principles are met, including clearly defined boundaries, rules adapted to local conditions, collective choice arrangements, monitoring, graduated sanctions, and recognized rights of self-organization (Ostrom, 1990). Applied to open source software, Ostrom's framework has been used to analyze how communities manage contributions, resolve conflicts, and sustain projects over time (Schweik & Kitsing, 2010). The sustainability of open source communities has been analyzed through a typology derived from Ostrom's principles, distinguishing between resource-based, infrastructural, and interactional dimensions of sustainability — and finding that the relationships among these dimensions are often characterized by tensions rather than simple synergies (Curto-Millet et al., 2022). However, unlike classical common pool resources, software is non-subtractable — one contributor's use does not diminish the available resource — which means that Ostrom's framework must be adapted rather than directly applied to the digital context (Curto-Millet et al., 2022; Schweik & Kitsing, 2010).

Platform economics — also known as the economics of two-sided markets and network effects — provides a third theoretical lens of particular relevance to the contemporary debate. The emergence of large cloud computing platforms has created a structural dynamic in which platform operators can appropriate enormous value from open source software without contributing proportionally to its development. By building proprietary managed services on top of open source infrastructure — as Amazon Web Services did with databases including MySQL and Elasticsearch — cloud providers capture the economic rents generated by the open source software's adoption while the original developers bear the maintenance costs (Bechara & Lechner, 2024; West & Gallagher, 2006). This dynamic illustrates the broader tension in platform economics between the platform operator, who captures

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

value through network effects and proprietary service layers, and the open source contributor communities whose labor creates the foundational value. It also helps explain the wave of relicensing decisions in the early 2020s, in which prominent open source companies shifted from OSI-approved licenses to more restrictive terms precisely to prevent this form of asymmetric value capture by cloud platforms (Bechara & Lechner, 2024).

A fourth theoretical lens of direct relevance is socio-technical systems theory, which holds that technology and social organization co-evolve and cannot be understood in isolation from one another (Trist & Bamforth, 1951; Orlikowski, 1992). Applied to the open source versus proprietary debate, this framework directs attention away from software as an artefact and toward the ensemble of technical components, organizational practices, governance structures, and user communities that together constitute a software 'system.' The open source model and the proprietary model, in this reading, are not merely different licensing arrangements but different socio-technical configurations — each embedding distinct assumptions about who contributes, who governs, and who bears risk. This perspective is consistent with Chesbrough's (2003) open innovation theory, which identifies the strategic value of permeable organizational boundaries for knowledge production, and with platform governance frameworks that analyze how rule-setting authority over digital infrastructures is distributed among platform owners, complementors, and users (Gawer & Cusumano, 2002). These frameworks are not applied mechanically in this article but inform the analytical vocabulary used throughout — particularly in Sections 7 and 9 — to interpret governance outcomes that purely economic or technical frameworks cannot adequately explain.

3.5 The Political and Geopolitical Dimension

The most recently theorized dimension of the open source versus proprietary debate is its political and geopolitical character. The concept of digital sovereignty — broadly understood as the capacity of a political entity to exercise autonomous control over its digital infrastructure, data, and technological capabilities — has risen rapidly to prominence in academic and policy discourse over the past decade, driven by a series of events that revealed the fragility of technological autonomy in an era of concentrated platform power (Pohle, 2024; Glasze et al., 2022).

The mechanism by which proprietary software can become a vector of technological dependence between nations is structural rather than conspiratorial. When a government, a corporation, or a national economy builds critical systems on proprietary platforms controlled by foreign actors, it implicitly accepts that the rights to modify, inspect, redistribute, or withdraw from those systems are not its own

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

(Biström et al., 2024). Dependencies of this kind are not merely commercial — they are strategic, because they create conditions under which a foreign state or corporation can exercise leverage by modifying pricing terms, withdrawing technical support, or, in extreme scenarios, denying access altogether (Bechara & Lechner, 2024). The Snowden revelations of 2013, which exposed the extent of U.S. intelligence surveillance conducted through and with the assistance of major proprietary technology platforms, served as a catalytic event that transformed digital sovereignty from a technical concept into a mainstream political priority, particularly in Europe (Pohle, 2024; Glasze et al., 2022).

Open source software has consequently been theorized and deployed as a strategic instrument of digital sovereignty, offering a potential pathway to technological autonomy through the elimination of the source code access problem that defines proprietary dependency. Countries in the Global South have increasingly adopted this framing: digital sovereignty initiatives reflect aspirations for greater autonomy from Silicon Valley-centered digital ecosystems, and open source infrastructure has been identified as a means of reclaiming agency and self-determination over digital technologies (Jiang et al., 2024). India's open-source Digital Public Infrastructure, built over the past decade to enable identification, financial inclusion, and e-governance for over one billion citizens, represents one of the most ambitious national deployments of this strategic logic (Jiang et al., 2024).

In Europe, the debate has been formally institutionalized. The European Commission has described digital sovereignty as Europe's ability to act independently in the digital world, and has framed reducing dependency on U.S. and Chinese proprietary technology as a central strategic objective (Pohle, 2024). The EU's GDPR, the AI Act, the Digital Markets Act, and the Digital Services Act together constitute a regulatory architecture that seeks to rebalance power between platform-controlling entities and users, citizens, and competing innovators — a project that intersects directly with the open source versus proprietary debate, since open source software is structurally more amenable to the regulatory transparency and auditability that these frameworks require (Glasze et al., 2022; Biström et al., 2024). Nonetheless, scholars have cautioned against treating digital sovereignty as a self-evidently coherent or uniformly progressive concept: its broad adaptability to diverse political contexts — from liberal democratic to authoritarian — means that it can equally serve as a justification for openness and for closure, for interoperability and for balkanization, depending on who invokes it and for what purpose (Pohle, 2024).

4. Historical Overview: How We Got Here

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

4.1 The Pre-Open Source Era: Software as Industrial Secret

To understand the present conflict between open source and proprietary software, it is necessary to trace the historical conditions that first produced proprietary software as a distinct economic category — for it was not always so. In the earliest decades of computing, software was not conceived of as an independent commercial product. It was provided as an accessory bundled with the hardware that manufacturers sold or leased, on the implicit understanding that the value resided in the machine itself and that software was a necessary but secondary instrument for making the hardware functional (Grad, 2002). During the 1950s and early 1960s, source code circulated relatively freely among university computing centers, research laboratories, and technically sophisticated corporate users, many of whom customized and shared programs among themselves as a matter of normal professional practice (Grad, 2002).

This arrangement was disrupted by a pivotal regulatory and commercial inflection point. In 1969, IBM began to unbundle software from mainframe hardware after the U.S. Department of Justice considered bundling to be an unfair selling practice (Ceruzzi, 2003; Grad, 2002). With this unbundling decision — announced on June 23, 1969, under direct pressure from pending antitrust litigation — IBM introduced the distinction between system control programs and program products, and the latter became a saleable commodity (Grad, 2002). Software changed, almost overnight, from a giveaway to a competitive commercial product, and this shift created the structural conditions under which an independent software industry could develop and grow (Grad, 2002). The government's antitrust suit against IBM, which lasted nearly thirteen years before being dropped in 1982, thus played a foundational role in shaping the political economy of software: by compelling the disaggregation of hardware and software markets, it indirectly enabled the emergence of firms like Microsoft, which would come to dominate the subsequent era of personal computing (Grad, 2002). Simultaneously, the late 1960s saw the growth of minicomputers providing an alternative software market, and by 1970 minicomputer unit shipments exceeded those of mainframes, further fragmenting the landscape and accelerating the commercialization of software development (Grad, 2002).

4.2 The Reaction: The Birth of the Open Source Movement

The proprietary software model, having been established through market forces and legal instruments in the 1970s and early 1980s, almost immediately generated a countervailing reaction. Richard Stallman's decision to launch the GNU Project in 1983 and found the Free Software Foundation in 1985 was a direct response to the commercial enclosure of software that had previously circulated freely in research

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

communities (Stallman, 1985; Tolu, 2018). The GNU Manifesto articulated a moral case for software freedom that framed proprietary licensing not merely as a commercial strategy but as an ethical violation of the principles of solidarity and mutual aid that Stallman believed should govern the technological commons. The General Public License (GPL), developed to protect GNU software, introduced a novel legal mechanism — copyleft — that used copyright law to enforce openness: any derivative work of GPL-licensed software had to be distributed under the same terms, thereby preventing the enclosure of open contributions within proprietary products (Stallman, 1985; Tolu, 2018).

A complementary and, in certain respects, parallel development occurred in Finland in 1991, when Linus Torvalds — then a twenty-one-year-old computer science student at the University of Helsinki — began writing a Unix-like kernel for his personal computer. Frustrated by the licensing restrictions of MINIX, a proprietary Unix variant, and unable to afford a full Unix license, Torvalds published his kernel publicly on August 25, 1991, inviting contributions from anyone willing to participate. What began as an explicitly hobby project rapidly attracted a global community of developers who extended, debugged, and improved the kernel through the distributed, peer-review model that Raymond would later theorize as the bazaar (Raymond, 1999). In January 1992, Torvalds relicensed the kernel under the GNU GPL — a decision he subsequently described as "definitely the best thing I ever did" — thereby aligning the Linux project with the broader free software ecosystem and enabling the combination of the Linux kernel with GNU tools to produce a complete, freely distributable operating system (Tolu, 2018).

The institutionalization of the movement took another step in 1998, when a group of practitioners and advocates — including Eric Raymond and Bruce Perens — coined the term "open source" and founded the Open Source Initiative precisely to make the practical case for collaborative development accessible to corporate audiences who found the explicitly political language of the Free Software Foundation off-putting (Tolu, 2018; O'Mahony, 2007). The rebranding was consequential: by framing openness in terms of software engineering efficiency and quality rather than political ethics, the OSI positioned open source as a legitimate and commercially attractive methodology, opening the door to corporate adoption at a scale the free software movement had been unable to achieve (Fitzgerald, 2006).

4.3 The Browser Wars and the Desktop Battle

The late 1990s produced the first high-profile, publicly visible confrontation between open and proprietary paradigms at the level of mass-market software, with consequences that reverberate to the present day. The Microsoft versus Netscape

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

browser war was at its core a struggle over which entity would control the user's primary interface to the World Wide Web — and thereby exercise gatekeeping power over the applications, commerce, and communication conducted through it (Hazlett, 2000). Netscape Communications had introduced Navigator in 1994 and by 1995 commanded approximately 80% of the browser market (West & Gallagher, 2006). Microsoft, recognizing the threat that a dominant, platform-independent browser represented to the strategic value of its Windows monopoly, launched Internet Explorer in August 1995 and proceeded to bundle it with Windows, negotiate exclusive pre-installation agreements with PC manufacturers, and design Windows to disadvantage competing browsers (Hazlett, 2000). The DOJ's theory, articulated in the antitrust suit filed in May 1998 against Microsoft by the Department of Justice and twenty U.S. states, was that without Microsoft's aggressive defensive reaction, Netscape Navigator combined with Sun's Java programming language could have provided a platform-independent middleware layer capable of eroding the significance of Windows as a prerequisite for application development — thereby opening the operating system market to genuine competition (Hazlett, 2000; Gilbert & Rubinfeld, 2001). Judge Thomas Penfield Jackson found in 2000 that Microsoft had unlawfully maintained its Windows monopoly and had unlawfully tied Internet Explorer to Windows, though the company ultimately avoided structural breakup under a settlement approved in 2002. Among the indirect consequences of Netscape's defeat was the founding of the Mozilla Foundation by former Netscape engineers, who released Firefox as an open source alternative browser — making the browser wars a foundational episode in the relationship between antitrust enforcement and the institutional ecology of open source software (West & Gallagher, 2006).

The concurrent battle between Linux and Windows for the desktop operating system market produced different lessons. While Linux achieved near-total dominance in server environments throughout the late 1990s and 2000s, its bid for the consumer desktop market did not succeed in displacing Windows or, later, macOS (Fitzgerald, 2006). The reasons were structural rather than merely technical: the proprietary software ecosystem surrounding Windows — particularly its dominance in productivity software, gaming, and enterprise applications — created powerful network effects and switching costs that made alternatives difficult to adopt even when technically competitive (West & Gallagher, 2006). The desktop "defeat" of Linux taught the open source community an important lesson about the relationship between technical quality and market adoption: superiority along engineering dimensions is insufficient when commercial ecosystems, vendor relationships, and user habits entrench incumbent platforms (Fitzgerald, 2006; Wen & Choudhury, 2019).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

4.4 The Cloud Turning Point: When Open Source Won Without Noticing

The emergence of the internet as a commercial infrastructure in the late 1990s and the subsequent construction of cloud computing in the 2000s represented a quiet but decisive victory for open source software, one that occurred largely outside the visibility of the desktop wars that had dominated public attention. The LAMP stack — Linux, Apache, MySQL, and PHP — became the de facto standard architecture for dynamic web applications beginning in the late 1990s. These four open source components, each independently developed and separately licensed, provided the foundational infrastructure for an enormous proportion of the commercial internet, from early content management systems through the scaling of social media platforms. The widespread adoption of LAMP demonstrated that open source components could be combined into coherent, production-grade architectures capable of supporting applications at arbitrary scale — dissolving the proprietary vendor's claim that only integrated, commercially supported software could be trusted for serious enterprise workloads (Fitzgerald, 2006; West & Gallagher, 2006).

The major technology corporations that built their businesses on this infrastructure — Google, Amazon, Meta, and Microsoft — simultaneously became the most significant beneficiaries and, paradoxically, the most significant contributors to open source software (Rikap, 2024). This relationship represents what scholars of open source business models have identified as the corporate open source paradox: firms strategically release code as open source precisely in order to commoditize components that competitors might otherwise use as proprietary differentiators, while retaining proprietary control over the service layers, data assets, and infrastructure where their competitive advantage actually resides (West & Gallagher, 2006; Widder et al., 2023). The edge is kept secret, whereas complementary or not so cutting-edge developments are frequently published, put in open source and/or patented — openness does not endanger appropriation because some pieces are always kept secret (Rikap, 2024). Google's release of TensorFlow and Meta's release of PyTorch as open source AI frameworks exemplify this dynamic: by making foundational AI infrastructure freely available, these firms attracted developer ecosystems that drove adoption of their broader platform services while foreclosing the possibility that smaller competitors might build proprietary differentiation at the infrastructure layer (ITIF, 2025).

The open core model emerged in this period as the dominant commercial structure for firms seeking to monetize open source software directly rather than indirectly through services and infrastructure (Riehle, 2009). Companies such as Red Hat — whose acquisition by IBM for $34 billion in 2019 marked a landmark in the commercial valuation of open source businesses — demonstrated that sustainable revenue could

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

be generated by providing enterprise-grade support, certification, and management tooling around freely available open source products (Fitzgerald, 2006). This model made the open versus proprietary distinction commercially negotiable: the same codebase could simultaneously serve open source community users and premium enterprise customers, with the licensing boundary carefully drawn to capture value without eliminating the community contributions that sustained the software's development (Riehle, 2009; Shah, 2006).

4.5 The Current Era: AI, LLMs, and the New Battle

The rapid diffusion of large language models (LLMs) since 2022 has opened a new and qualitatively distinct front in the open source versus proprietary debate, one that combines the earlier conflicts over code accessibility with novel questions about training data transparency, computational resource requirements, and the governance of systems capable of consequential autonomous action (Widder et al., 2023; European Commission, 2025).

The first major salvo in this new conflict was fired in February 2023, when Meta released the weights of its LLaMA model to selected researchers — making it the first organization to make a large foundation model available for examination and modification outside the bounds of a proprietary API (ACLU, 2025). The subsequent July 2023 release of Llama 2 under a commercial license, making it freely available to enterprises and developers, transformed the landscape: within months, the open source community had produced thousands of derivative fine-tuned models on the Hugging Face platform. The release of Llama 3.1 in July 2024 — including a 405 billion parameter variant that achieved performance competitive with GPT-4, Google's Gemini, and other leading proprietary systems — marked the first time an open-weight model had reached frontier-level capability (Meta AI, 2024). Since 2022, the number of publicly released AI models has more than doubled, and more of them now come with open weights, giving developers far greater transparency and control, while open models are quickly closing the gap with proprietary systems (European Commission, 2025).

The governance literature on open source AI remains nascent and contested. Mökander et al. (2023) argue that transparency in AI systems requires disclosure beyond weights to include training procedures and evaluation protocols, while Solaiman (2023) identifies a spectrum of release practices that resist binary open/closed classification. These positions are in tension with Kapoor & Narayanan (2023), who document that most 'open' model releases fall short of reproducibility standards, and with Henderson et al. (2023), who raise questions about whether

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

openness in AI exacerbates rather than mitigates misuse risks — a debate that remains empirically unresolved.

Yet a critical definitional controversy has accompanied this apparent triumph of openness. The term "open source" as applied to AI models such as Llama is disputed by the open source community itself on grounds that go to the heart of what openness means in an era of data-dependent, compute-intensive AI systems. A November 2024 article in *Nature* asserted that describing Llama 3 as "open" constitutes "openwashing" — applying the vocabulary of openness to systems that provide little more than an API or the ability to download a model subject to distinctly non-open use restrictions. Researchers at Radboud University scored Llama 2 with the second-lowest "openness" ranking in a comparative study of twenty LLMs, criticizing Meta's use of the term open source as "positively misleading" on grounds that the training data is entirely undocumented and the technical documentation is inadequate. The key distinction is between open weights — which allow inspection and deployment of a trained model — and the far more demanding standard of full open source, which would require disclosure of training data, training code, data pipelines, and evaluation methodology. As noted by Widder et al. (2023) and the Open Source Initiative (2024), open weights without training data transparency do not enable the independent reproduction, full auditability, or community-governed development that classical open source software entails (Widder et al., 2023; ACLU, 2025).

The geopolitical dimension of AI has further transformed this debate. National AI sovereignty strategies have proliferated across regions with differing orientations toward open and closed models. India's National AI Stack, proposed as a seven-layer framework covering hardware, compute, data, model, API, and security layers, is designed to reduce reliance on foreign AI frameworks by fostering indigenous AI development across critical sectors (Srivastava & Mishra, 2025). Brazil's AI sovereignty model is rooted in co-creation between public institutions, universities, and domestic firms, supporting Portuguese-language models including SoberanIA, Amazônia 360, and GovBERT-BR under its AI Plan 2024–2028, with approximately four billion dollars in committed public R&D investment (Institute for Global Change, 2026). The European Commission's report on the European open source AI landscape identifies open source AI as a lever for both competitiveness and digital sovereignty, noting that over half of developers regularly rely on open models, datasets, and tools, and that compute access — addressed through nineteen EU-funded AI Factories and expanded EuroHPC supercomputers — remains the primary structural barrier to European open source AI development (European Commission, 2025). Scholars analyzing EU AI policy have identified a persistent tension between what they term jurisdictional sovereignty — the relative independence of the EU from major foreign powers — and citizen sovereignty, defined as protection of individuals

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

from large technology companies regardless of their national origin (Mügge, 2024). These divergent conceptions of sovereignty produce genuinely different policy prescriptions, and neither fully resolves the open source versus proprietary dilemma: open source AI can serve both democratic and authoritarian sovereignty projects, and the mere availability of model weights is no guarantee of meaningful citizen control over AI systems that shape public life (Mügge, 2024; Widder et al., 2023).

5. Technical Components: What Is at Stake

5.1 Operating Systems

5.1.1 Linux and Its Distributions

Linux and its proliferation of distributions — including the community-governed Debian, the commercially sponsored Ubuntu, the community-governed Fedora, and the commercially licensed Red Hat Enterprise Linux — constitute the most successful deployment of open source software in the history of computing. Linux powers all 500 of the world's fastest supercomputers and has maintained that position without interruption since November 2017, reflecting its scalability, modularity, and capacity to accommodate customization at every layer of the stack (Pfandzelter & Bermbach, 2024). The distribution ecosystem itself illustrates the diversity of governance models that open source can accommodate: Debian is maintained by a volunteer community under a formal constitution, Ubuntu is produced commercially by Canonical Ltd. on an open source foundation, and RHEL is a proprietary enterprise product built from the openly available Fedora source base — with its upstream code made freely available as CentOS Stream (Fitzgerald, 2006).

Table 1. Operating System Market Share by Infrastructure Layer (observational data; interpretation follows in text):

Infrastructure Layer	Linux / OSS (%)	Windows / Proprietary (%)	Source
Top 500 Supercomputers	100.0	0.0	TOP500, Nov 2024
Public Cloud Workloads	>90.0	<10.0	XtendedView, 2025
Web Servers (Top	96.3	3.7	XtendedView,

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

1M sites)			2025
Desktop / Laptop (Consumer)	4.3	88.2 (Win+macOS)	StatCounter, Feb 2025
Container Orchestration (prod.)	80.0	n/a (OSS standard)	CNCF Survey, 2025

Interpretive note: the data above are observational and reflect adoption rates, not controlled performance experiments. The divergence between infrastructure dominance (+95 pp over Windows in cloud) and consumer desktop marginality (4.3%) reflects economic and ecosystem factors analyzed in Section 9.1, not technical inferiority of either model.

Android presents the most commercially consequential and most contested case study in the relationship between Linux and proprietary software. At its core, the Android Open Source Project (AOSP) is licensed under the Apache License 2.0 and makes its source code freely available. However, most commercially shipped Android devices run with a substantial proprietary layer on top of AOSP: Google Mobile Services, which includes the Play Store, Play Services, Chrome, and cloud messaging infrastructure, is not part of AOSP and must be licensed separately from Google (Mayrhofer et al., 2021). Android's source code does not contain the device drivers, often proprietary, that are needed for certain hardware components, nor the source code of Google Play Services, which many applications depend upon; as a result, most Android devices ship with a combination of free, open source, and proprietary software (Mayrhofer et al., 2021). In March 2025, Google announced that Android OS development would be shifted entirely to its internal branch, with source code published to AOSP only after each major release — a decision that further reduces the community governance dimension of the project. This configuration represents precisely the open core dynamic described in Section 2: the open source AOSP layer commoditizes the platform and attracts an OEM ecosystem, while the proprietary GMS service layer captures the commercial value (West & Gallagher, 2006; Riehle, 2009).

5.1.2 Windows and macOS

Despite Linux's dominance in server and cloud environments, Microsoft Windows maintains a commanding position in the enterprise desktop and productivity software market, and macOS holds a significant share of the professional creative and developer desktop market (Wen & Choudhury, 2019). The persistence of this desktop dominance despite open source alternatives reflects structural factors: the ecosystem

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

lock-in created by decades of proprietary application development, the organizational inertia of enterprise IT procurement, and the network effects of dominant software platforms (West & Gallagher, 2006).

Microsoft's strategic shift toward open source has been one of the most significant corporate realignments in the software industry since the mid-2010s. The Windows Subsystem for Linux, introduced in 2016 and substantially enhanced in subsequent versions, allows Windows users to run a full Linux kernel environment natively within Windows — a development that reflects pragmatic accommodation to developer workflows increasingly dependent on Linux tooling (Bechara & Lechner, 2024). Microsoft's acquisition of GitHub in 2018, its development of the open source VS Code editor, and its contributions to the Linux kernel represent a substantive change in posture from the company that, in the early 2000s, characterized open source as a threat to intellectual property (Fitzgerald, 2006). This transformation is better understood as strategic rather than ideological: by contributing to open source infrastructure upon which developers depend, Microsoft strengthens the attractiveness of its cloud and enterprise services, which remain proprietary (West & Gallagher, 2006; Rikap, 2024).

Apple's macOS similarly occupies a hybrid position: its kernel is derived from the open source Mach and BSD Unix projects, and Apple contributes to open source projects including WebKit and LLVM. However, the macOS user experience and application ecosystem constitute a rigorously managed proprietary environment in which Apple exercises direct control over hardware design, software signing requirements, and distribution through the App Store — a configuration that critics of Apple's platform governance argue restricts user freedom and downstream competition (Shah, 2006; Bonaccorsi & Rossi, 2003).

5.1.3 Embedded Systems and IoT

In the domain of embedded systems and the Internet of Things, open source operating systems — primarily Linux — are overwhelmingly prevalent. The wide deployment of Linux in IoT devices has introduced a distinct category of security risk: many manufacturers deploy customized Linux-based firmware without maintaining the patch cadences necessary to address disclosed vulnerabilities, effectively creating a large population of devices running open source software that nobody is actively auditing or updating (Bechara & Lechner, 2024; Biström et al., 2024). This configuration inverts the open source security argument: the theoretical benefit of open code — that anyone can audit and fix vulnerabilities — is realized only when the governance structures and resource commitments to actually perform that auditing are in place (Walden, 2020).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

5.2 Databases

5.21 Open Source Databases

The database market provides one of the clearest illustrations of open source's structural victory at the infrastructure layer combined with persistent proprietary dominance at the premium enterprise tier. PostgreSQL, originally developed at UC Berkeley and released under the permissive PostgreSQL License, has achieved widespread adoption as the leading choice in developer surveys, with its governance model — maintained by a global volunteer community with no single corporate owner — providing long-term architectural predictability that is frequently cited as a reason for trust among enterprise adopters (Curto-Millet et al., 2022).

The MySQL/MariaDB bifurcation constitutes one of the most instructive governance episodes in open source database history. When Oracle acquired Sun Microsystems in 2010 and thereby gained stewardship of MySQL, MySQL's founding developer forked the codebase on the day of the acquisition announcement to create MariaDB, which was placed under the governance of a non-profit foundation — driven by concern that Oracle, whose commercial database product competed directly with MySQL, would not serve as an adequate steward of the open source project (O'Mahony, 2007). The fork is consistent with the interpretation that open source governance is not a technical but a political and economic question: the right to fork is a real protection against adverse corporate stewardship, but exercising it fragments the ecosystem and imposes transition costs on users (O'Mahony, 2007; Shah, 2006).

The NoSQL segment — including MongoDB, Redis, and Apache Cassandra — presents a further complication of the open source versus proprietary classification. Many NoSQL databases were originally developed under permissive open source licenses and subsequently relicensed under more restrictive terms in direct response to asymmetric value capture by cloud providers: MongoDB moved to the Server Side Public License in 2018; Redis moved to a dual licensing model in the same period; both decisions represent responses to cloud providers offering managed services based on these databases without contributing to their development (Bechara & Lechner, 2024; West & Gallagher, 2006).

5.2.2 Proprietary Databases

Oracle Database retains dominant penetration in large enterprise and mission-critical segments — particularly in financial services, telecommunications, and government — where decades of investment in Oracle-specific stored procedures, features, and certifications create extremely high switching costs (West & Gallagher, 2006).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Microsoft SQL Server similarly leverages deep integration with the broader Microsoft enterprise stack as a competitive moat that makes substitution with open source alternatives technically feasible but organizationally disruptive (Petcu et al., 2013). The most significant contemporary development in the proprietary database segment is the emergence of cloud-native managed database services — Amazon Aurora, Google Spanner, and Microsoft Azure SQL Hyperscale — which layer proprietary service architecture and management capabilities on top of open source database engines, extracting economic rents from open source adoption while retaining service layers as competitive differentiators (Bechara & Lechner, 2024).

5.2.3 The License Wars

The conflict over database licenses reflects a broader structural tension: the original permissive and copyleft licenses were designed for a world of software distribution, not a world of cloud services. The Commons Clause and SSPL represent industry responses to this failure — attempts to prohibit the specific commercial behavior of providing an unmodified open source product as a cloud service that original open source licenses did not anticipate. Both are explicitly rejected by the Open Source Initiative as failing to meet the Open Source Definition, meaning that projects adopting them are no longer, by OSI standards, open source (Riehle, 2009). This creates a dilemma for project communities: protect the project economically at the cost of its open source identity, or maintain the open source principle at the cost of asymmetric commercial exploitation (West & Gallagher, 2006; Bechara & Lechner, 2024).

5.3 Programming Languages and Runtimes

5.3.1 Open Languages

The dominant general-purpose programming languages of the present era are predominantly open source and community governed. Python, administered by the Python Software Foundation under a permissive license, has become the lingua franca of data science, machine learning, and scientific computing — with its governance model providing a degree of long-term stability that proprietary language vendors have historically struggled to match (Lerner & Tirole, 2002). Rust, developed initially by Mozilla and now stewarded by the independent Rust Foundation, has attracted significant systems programming adoption on the basis of its memory safety guarantees; the Linux kernel began accepting Rust code contributions in 2022, a milestone that reflects both the language's technical maturity and the open source governance model that made such cross-project integration possible (Fitzgerald, 2006). Go, developed by Google and released under a BSD-style license, illustrates the strategic corporate open source model: Google benefits from community adoption

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

that extends its developer ecosystem while retaining effective influence over language evolution through its role as the primary steward of the reference implementation (Rikap, 2024).

JavaScript's governance presents a more complex case. The language specification is maintained by Ecma International's TC39 committee. The Node.js ecosystem has been a site of governance conflict: the io.js fork in 2014–2015 was driven by community dissatisfaction with Joyent's stewardship, and the subsequent donation of Node.js to the OpenJS Foundation was a direct response to that governance crisis (O'Mahony, 2007). The npm package registry — which hosts the open source modules that constitute the effective ecosystem of Node.js development — was acquired by GitHub and thereby by Microsoft in 2020, concentrating control over a critical infrastructure component of the open source web development ecosystem in a single corporation (West & Gallagher, 2006).

5.3.2 Proprietary and Semi-Open Languages

Several important programming languages occupy a position between fully open and fully proprietary. Swift, developed by Apple and released as open source in 2015, is technically governed through an open process but designed primarily for Apple platform development — creating a form of strategic openness that expands the developer community for Apple's ecosystem without enabling genuine competition at the platform level (Shah, 2006). Microsoft's .NET platform represents perhaps the most significant transformation in language openness: originally a fully proprietary ecosystem, Microsoft released .NET Core as open source in 2016 and subsequently open-sourced much of the remaining .NET stack, driven by competitive pressure from open source alternatives and the strategic imperative to attract developers across non-Windows platforms (Fitzgerald, 2006; Bechara & Lechner, 2024).

5.3.3 Implications for Technical Lock-In

The programming language ecosystem illustrates how technical lock-in operates in layers. Even when a language is itself open source, the ecosystem of frameworks, libraries, tooling, and platform integrations built around it can create switching costs that effectively constrain organizations to a particular vendor's infrastructure (Petcu et al., 2013). An organization deeply invested in Azure-specific .NET services faces migration costs that are not resolved by the open sourcing of the language runtime; an organization that builds its application logic in Swift is functionally dependent on Apple platforms regardless of Swift's license. This fragmentation of ecosystems — each centered on a different corporate sponsor with a proprietary service layer above the open source foundation — represents a structural challenge for interoperability

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

and portability that open source licensing alone does not address (Biström et al., 2024; Lenk et al., 2011).

5.4 Cloud Platforms and Infrastructure

5.4.1 Open Source Infrastructure

The cloud native DevOps infrastructure stack has become overwhelmingly dominated by open source technologies. Kubernetes, originally developed by Google and donated to the Cloud Native Computing Foundation in 2015, has achieved near-universal adoption as the standard for container orchestration: production use reached 80% of surveyed organizations in 2024, up from 66% in 2023, with 93% of organizations using, piloting, or evaluating the platform (CNCF, 2025). Docker, Terraform, Helm, Prometheus, and Argo collectively constitute an industry-standard open source infrastructure stack for application deployment and observability, maintained through a governance model in which multiple competing companies contribute to shared foundations while differentiating at the service layer (CNCF, 2025). The CNCF plays an institutional role: it provides governance frameworks for projects to achieve neutrality from any single corporate sponsor, graduation criteria signaling project maturity, and a legal structure that manages intellectual property contributions (CNCF, 2025).

OpenStack, launched in 2010 as an open source alternative to AWS that would allow organizations to build their own infrastructure-as-a-service environments, represents the most ambitious and instructive failure in the history of open source cloud infrastructure. Despite substantial corporate backing, OpenStack struggled with architectural complexity, inconsistent documentation, and the difficulty of deployment and maintenance at scale without significant specialized expertise — ultimately ceding the market it sought to democratize to the proprietary clouds it was designed to challenge (West & Gallagher, 2006; Fitzgerald, 2006). The OpenStack episode teaches a fundamental lesson about the limits of open source as a market intervention strategy: technical availability does not overcome the organizational and economic advantages of a vertically integrated proprietary service provider with massive capital investment in reliability engineering.

5.4.2 Proprietary Clouds

Amazon Web Services, Microsoft Azure, and Google Cloud collectively control the overwhelming majority of the global public cloud market, and each derives substantial competitive differentiation from proprietary managed services built on top of open source infrastructure (Rikap, 2024). The open source versus proprietary distinction in

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

this context plays out not at the infrastructure layer — where Linux and Kubernetes are universal — but at the service layer: managed APIs, data services, and machine learning platforms that cloud providers offer as premium services are typically proprietary, creating the vendor lock-in dynamics that constitute one of the most significant contemporary concerns about cloud dependency (Bechara & Lechner, 2024).

Vendor lock-in in cloud environments operates through multiple reinforcing mechanisms. Technical lock-in occurs when a cloud service provider offers proprietary technologies, APIs, or data formats that are not easily transferable to other providers, making it challenging to migrate without significant redevelopment effort (Alhosban et al., 2024). Data lock-in arises when storage formats and structures are owned by the cloud provider, making it difficult to move structured data to another service provider or infrastructure (Alhosban et al., 2024). Research involving surveys of 114 organizations found that customers are largely unaware of proprietary standards that prohibit interoperability and portability when procuring services from vendors, and that the complexity and cost of switching providers is consistently underappreciated until implementation (Linthicum, 2016). The true cost of vendor lock-in extends beyond direct financial charges to encompass technical entrenchment via proprietary service dependencies and organizational inertia caused by workforce specialization in vendor-specific tooling (Wen & Choudhury, 2019).

5.4.3 Data Sovereignty and National Clouds

The cloud platform landscape has become a primary arena for digital sovereignty initiatives. The European Gaia-X project, launched in 2019 with backing from the European Commission and major European technology firms, was conceived as a federated data infrastructure enabling European organizations to store and process data under European legal jurisdiction using interoperable standards — with open source positioned as a prerequisite for its sovereignty claims, since only systems whose code can be independently audited can provide the transparency guarantees necessary for genuine data sovereignty (Glasze et al., 2022; Biström et al., 2024). However, Gaia-X has encountered significant implementation difficulties, with critics arguing that the participation of U.S. cloud providers as members of its governance structure undermines its sovereignty objectives (Pohle, 2024). Brazil's national cloud strategy similarly invokes open source infrastructure as a mechanism for reducing strategic dependency on foreign proprietary platforms, with an emphasis on indigenous development capacity rather than mere procurement of domestically hosted foreign software (Institute for Global Change, 2026).

5.5 Development Tools and IDEs

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

5.5.1 Open Source Development Environments

Visual Studio Code, released by Microsoft under the MIT License, has become the world's most widely used code editor. Its success exemplifies the strategic open source paradox: the editor itself is open source, but it is deeply integrated with Microsoft's proprietary infrastructure through its default telemetry collection, the Visual Studio Marketplace extensions ecosystem, and its tight integration with GitHub Copilot — all of which serve Microsoft's commercial interests without undermining the open source license (Rikap, 2024; West & Gallagher, 2006). The fully open source VSCodium distribution, which strips Microsoft's telemetry and proprietary binaries, exists precisely because the MIT license permits it — but commands only a fraction of VS Code's user base, illustrating how open source licensing and open source governance can diverge substantially in practice.

Vim and Emacs represent a philosophically distinctive tradition within the development tool ecosystem. Emacs, maintained under the GNU GPL and governed by the Free Software Foundation, embodies Stallman's vision of the editor as a programmable, user-modifiable environment — a platform for the exercise of software freedom rather than merely a productivity tool (Stallman, 1985). Their persistence despite the emergence of far more feature-rich commercial and semi-commercial alternatives attests to the continuing appeal of transparency and user control as values independent of functional performance metrics.

5.5.2 Proprietary Development Tools

JetBrains' IDE suite — IntelliJ IDEA, PyCharm, WebStorm, and associated products — represents the leading commercial alternative in the professional developer tooling market. Operating on a subscription model, JetBrains has maintained a premium position by investing heavily in language intelligence features — code completion, refactoring, and static analysis — that open source alternatives have historically struggled to match (Bonaccorsi & Rossi, 2003). Xcode, Apple's proprietary integrated development environment, functions as a gatekeeper for iOS and macOS development: not only is Xcode required for building and signing applications for Apple's platforms, but it is available exclusively on macOS, creating a hardware lock-in that reinforces Apple's ecosystem control and forces developers seeking to build iOS applications to own Apple hardware (Shah, 2006).

5.5.3 The New Front: AI-Assisted Development

The emergence of AI-powered code completion and generation tools has opened a new dimension of the open source versus proprietary debate within the development

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

tooling domain. GitHub Copilot, developed by GitHub and Microsoft using OpenAI's language models, is a proprietary subscription service that provides AI code completion integrated into VS Code and other editors. Copilot has been the subject of academic and legal controversy: its training on public GitHub repositories has raised questions about whether generated code may reproduce fragments of GPL-licensed code in contexts that violate the GPL's terms — a question that remains unresolved in courts (Widder et al., 2023). The open source response includes tools such as Continue.dev, which integrates with locally deployed open-weight language models via Ollama, allowing developers to use AI code assistance on sensitive codebases without routing prompts through a third-party proprietary API (ACLU, 2025). The emergence of AI-native editors that combine open source code editing interfaces with proprietary AI inference backends illustrates the same open core dynamic identified throughout this analysis: open source at the user-facing layer, proprietary control at the intelligence layer where competitive differentiation resides (Widder et al., 2023).

5.6 Artificial Intelligence and Language Models

5.6.1 Proprietary AI Models

The leading frontier AI systems — OpenAI's GPT series, Anthropic's Claude, and Google's Gemini — are proprietary systems accessible exclusively through API interfaces, with no disclosure of model weights, training data, or training methodologies (Widder et al., 2023). The risks from a user and societal perspective are equally substantial: dependency on a proprietary API for critical applications creates a single point of commercial failure; the opacity of model behavior makes independent safety evaluation impossible; and the concentration of frontier AI capability in a small number of large corporate entities raises structural concerns about the alignment of AI development trajectories with public interests (Widder et al., 2023; European Commission, 2025).

5.6.2 Open Source and Open-Weight AI Models

The open-weight AI model ecosystem has expanded dramatically since Meta's release of the first LLaMA models in 2023, with Mistral, Falcon, and Microsoft's Phi series providing a growing range of capable models available for local deployment, fine-tuning, and inspection (ACLU, 2025; European Commission, 2025). The primary practical advantages of open-weight models over proprietary API-only systems include: the ability to run models locally without transmitting sensitive data to third-party services; freedom to fine-tune models on domain-specific data without restriction; cost advantages at scale from performing inference on owned hardware;

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

and the ability to inspect and modify model behavior in ways that closed API access does not permit (Widder et al., 2023). However, as analyzed in Section 4.5, the application of the term "open source" to open-weight models is contested: open weights without disclosure of training data, training code, and evaluation methodology do not enable the full community governance and independent reproduction that characterize classical open source software (Widder et al., 2023; ACLU, 2025).

A small-scale empirical evaluation conducted as part of this study corroborates this convergence at the task level. Across 120 structured prompts covering mathematical reasoning, logical inference, and general knowledge retrieval, the open-weight model achieved 91.7% accuracy compared to 93.3% for the proprietary model — a difference of 1.6 percentage points. Response latency averaged 3.50s for the open-weight model versus 1.79s for the proprietary model (a difference of 1.71s, with the open-weight model being slower). Output consistency across repeated runs was 91.7% for the open-weight model compared to 93.3% for the proprietary model, again reflecting a difference of 1.6 percentage points. These results are indicative rather than conclusive, given the limited sample size and task scope; they are reported here for transparency and should not be generalized beyond the specific models and task categories tested (see Supplementary Material S1 at https://github.com/anthony-santos-batista/ai-benchmark-study).

Table 3. Micro-Benchmark Summary: Open-Weight vs Proprietary Model (n = 120 prompts)

Metric	Open-Weight Model	Proprietary Model	Delta
Accuracy (% correct)	91.7%	93.3%	−1.6 pp
Avg. Response Latency (s)	3.50	1.79	+1.71s (open slower)
Output Consistency (%)	91.7%	93.3%	−1.6 pp

Note: Results are indicative only. Sample size (n = 120) and task scope are insufficient for generalization. Full prompt set and raw outputs available in Supplementary Material S1.

5.6.3 AI Infrastructure

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

The infrastructure layer of the AI ecosystem is predominantly open source. PyTorch, developed by Meta AI Research and donated to the Linux Foundation as the PyTorch Foundation in 2022, has become the dominant framework for AI model development and research (European Commission, 2025). Hugging Face has emerged as the central platform for the open source AI ecosystem: it hosts model weights, datasets, and training code for thousands of models, and provides a software library that has become the standard API layer for working with language models across frameworks (European Commission, 2025). However, Hugging Face's central role introduces a structural concentration risk: a single private company now controls the primary distribution infrastructure for open source AI, and its commercial decisions directly affect the entire open source AI ecosystem (Widder et al., 2023). The inference serving stack — including ONNX, vLLM, and llama.cpp — enables open-weight models to be deployed efficiently on diverse hardware, including consumer-grade GPUs and CPUs, a critical enabler of local deployment and organizational sovereignty over AI workloads (ACLU, 2025).

5.6.4 Governance and Ethics in Open Source AI

The governance of open source AI introduces ethical dimensions that do not arise in the same form for open source software. When an open-weight AI model is fine-tuned to produce harmful content or to facilitate malicious use cases, the questions of responsibility and mitigation are considerably less well-defined than for software vulnerabilities (Widder et al., 2023). The Open Source Initiative published its official definition of Open Source AI in October 2024, requiring at minimum that an Open Source AI system provide the model weights under an OSI-approved license, the source code used to train and run the model, and sufficient data information to replicate or retrain the system — a definition that most currently labeled "open source" AI systems do not fully satisfy (Widder et al., 2023; European Commission, 2025).

5.7 Security and Cryptography

5.7.1 The Case for Open Source in Security

The debate between "security through obscurity" and the open source security paradigm has been largely resolved in favor of the latter by both theoretical argument and empirical evidence (Anderson, 2020). Kerckhoffs's principle holds that a system should be secure even if everything about it — except the key — is public knowledge, making obscurity of implementation an insufficient security control (Anderson, 2020). OpenSSL and LibreSSL constitute the cryptographic infrastructure underlying the majority of encrypted internet traffic; their status as open source code that any security researcher can audit and contribute to represents a genuine public good —

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

though, as Heartbleed demonstrated, the theoretical availability of code for review is not equivalent to its actual review (Walden, 2020).

5.7.2 The Case for Proprietary Software in Security

The proprietary model's security argument rests not on the superior quality of closed code, but on the different risk-management structures it enables. Some enterprise security products derive value from threat intelligence that must remain private to prevent adversaries from adapting to it, making opacity a genuine operational security requirement (West & Gallagher, 2006). Commercial security vendors offer capabilities that community-governed open source projects structurally cannot easily match: legally binding service level agreements for critical vulnerability response, dedicated security engineering teams, and formal certifications — including FIPS 140-2/3 for cryptographic modules and FedRAMP authorizations for U.S. government cloud services — that require controlled software builds incompatible with distributed community release processes (Bonaccorsi & Rossi, 2003; Petcu et al., 2013).

5.7.3 Landmark Vulnerabilities and Their Lessons

Four high-profile security incidents from 2014 to 2024 provide a rich empirical basis for evaluating the security claims of both models.

The Heartbleed vulnerability (CVE-2014-0160), disclosed in April 2014, exposed a critical flaw in the OpenSSL Heartbeat extension — the vulnerability allowed attackers to remotely dump protected memory, including private cryptographic keys, from an estimated 24–55% of popular HTTPS sites at the time of disclosure (Durumeric et al., 2014). A boundary-checking error introduced by a contributor in December 2011 had gone undetected for over two years despite the theoretical availability of the code for public review. Internet-wide measurement studies found that over 73% of vulnerable certificates had not been reissued and over 87% had not been revoked three weeks after Heartbleed's public disclosure, revealing the gap between the theoretical security benefits of open code and the actual organizational capacity to respond to disclosed vulnerabilities (Zeber et al., 2015). Heartbleed's critical lesson was not that open source is insecure, but that the security benefits of open code are conditional on the existence of actual, resourced review: the OpenSSL project had, prior to Heartbleed, been maintained with effectively zero security review capacity despite serving as the cryptographic infrastructure of the global web (Walden, 2020). Following Heartbleed, the Core Infrastructure Initiative — subsequently succeeded by the Open Source Security Foundation — was established to provide sustained funding for critical open source security projects (OpenSSF, 2024).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

The SolarWinds attack (2020) represents the equally instructive proprietary software counterpoint. Malicious actors gained access to the source code of SolarWinds' Orion network monitoring platform — a proprietary, commercially licensed product — and inserted the SUNBURST backdoor into a legitimate software update distributed to approximately 18,000 customers, including multiple U.S. federal agencies (Ghanbari et al., 2024; Massacci et al., 2021). The attack was undetected for approximately nine months, demonstrating that proprietary code — because its development process is closed to external review — can be compromised at source and the compromise distributed to customers who have no means of independently verifying the integrity of what they receive (Ghanbari et al., 2024). Research directions in software supply chain security now identify the SolarWinds build-process compromise as exemplary of an attack vector in which the tooling and automation that turns code into a production application is itself targeted — a vector that is not unique to either open or proprietary software but that is particularly difficult to detect in closed development environments (Williams & Zacchiroli, 2025).

Log4Shell (CVE-2021-44228), disclosed in November 2021, demonstrated the invisible dependency problem in open source ecosystems. Log4j, a widely used Java logging framework maintained by the Apache Software Foundation, was embedded as a transitive dependency in a vast number of enterprise applications whose operators were unaware of its presence until the vulnerability was disclosed — at which point frantic auditing revealed how deeply Log4j had permeated the software supply chain (Ghanbari et al., 2024; Williams & Zacchiroli, 2025). Log4Shell's lesson concerns the nature of open source dependency management: the ubiquity of open source components in enterprise software creates a supply chain exposure that is in some respects structurally analogous to the proprietary supply chain exposure demonstrated by SolarWinds, though the transparency of open source code enables detection and remediation that closed-source products cannot as readily offer.

The XZ Utils backdoor (CVE-2024-3094), discovered in March 2024, represents the most sophisticated and alarming open source security incident of recent years. An unknown threat actor spent approximately 2.6 years systematically building credibility as a contributor to the XZ Utils compression library — a dependency of systemd in major Linux distributions — through a combination of legitimate technical contributions and apparent social engineering that applied pressure on the project's sole overworked maintainer, eventually gaining co-maintainer status and introducing a sophisticated backdoor in the February 2024 release tarballs (Pirozzi et al., 2025). The attack was discovered by chance — by a Microsoft software engineer noticing anomalous CPU usage during SSH logins — before the compromised versions had been incorporated into any stable major Linux distribution (Pirozzi et al., 2025). Analysis of the attack reveals a new category of supply chain threat that manipulates

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

not code but the software engineering practices themselves — from community management to CI/CD configurations — to establish legitimacy and maintain long-term control (Pirozzi et al., 2025). The XZ Utils attack revealed a structural governance vulnerability of open source projects that depend on the sustained, unpaid labor of individual maintainers without institutional support: the combination of maintainer burnout and community trust mechanisms creates an attack surface that is a human and governance vulnerability, not merely a technical one (Pirozzi et al., 2025; OpenSSF, 2024).

Table 2. Comparative Security Metrics: Four Landmark Incidents 2014–2024 (empirical data; interpretation follows in text)

Incident	Model	CVSS Score	Undetected Period	Scope	Auditability
Heartbleed (CVE-2014-0160)	Open source	7.5 (Critical)	~2 years (Dec 2011–Apr 2014)	24–55% of HTTPS sites; 73% certs not reissued at 3 wks	High (public source); review capacity near zero
SolarWinds /SUNBURST (2020)	Proprietary	N/A (supply chain)	~9 months	~18,000 customers incl. U.S. federal agencies	None (closed source; build compromise undetectable)
Log4Shell (CVE-2021-44228)	Open source	10.0 (Critical)	Public exploit within hours of disclosure	Widespread transitive dependency in enterprise Java stack	High (public source); patch issued within days
XZ Utils (CVE-2024-3094)	Open source	10.0 (Critical)	~2.6 years (attacker build-up)	Intercepted before stable distro inclusion; 0 production systems confirmed compromised	High (public source); discovered by anomalous performance observation

Interpretive note: the four incidents do not support a simple conclusion that one model is more secure. The pattern indicates that open source incidents are detectable and

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

patchable faster, while proprietary incidents benefit from a closed attack surface but lack independent verification. The relevant comparison is between failure modes, not aggregate rates.

6. Economic Dimensions

6.1 Open Source Business Models

The commercial viability of open source software was, for much of the movement's early history, treated as a paradox: how can a firm generate sustainable revenue from software that it gives away for free? The past two decades have conclusively resolved this paradox in practice, even if its theoretical explanation remains contested. A taxonomy of open source business models derived from the analysis of 120 commercial open source companies identifies seven archetypal patterns — open source platform, funding-based, infrastructure, open innovation, open core, proprietary-like, and traditional OSS business models — reflecting the wide range of strategies through which firms have learned to create and capture value in the presence of freely available source code (Weiss et al., 2022).

6.1.1 Support and Services: Red Hat / IBM as the Paradigmatic Case

The services and support model was the first commercially successful open source business model and remains, in its most mature form, the paradigmatic case for the entire category. Red Hat, founded in 1993, built its business not on the sale of software but on the sale of enterprise-grade support, certification, training, and guaranteed patch cadences for distributions of freely available open source software — primarily Linux (Lakhani & von Hippel, 2003). The company's core value proposition was straightforward: the software is free, but the reliability engineering, professional services infrastructure, and contractual accountability that enterprise customers require to trust it in production environments are not. This model demonstrated that open source software, far from being incompatible with commercial sustainability, could generate substantial and recurring enterprise revenue — Red Hat's fiscal year 2019 revenue was $3.4 billion, growing at 15% year over year, at the time IBM acquired the company for approximately $34 billion in the largest software acquisition in history (IBM, 2019). Red Hat's pioneering business model helped bring open source technologies, including Linux and Kubernetes, into the mainstream for enterprises — validation, as scholars of open source commercialization have noted, that community-driven innovation can generate value

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

commensurate with or exceeding proprietary alternatives (Dahlander & Magnusson, 2008; West & Gallagher, 2006).

6.1.2 Open Core: GitLab, HashiCorp, Elastic

The open core model — in which a firm releases a feature-limited version of its software under an OSI-approved license while commercializing an enterprise-grade version under a proprietary commercial license — has become the dominant structural form for venture-backed open source companies in the past decade (Riehle, 2009). GitLab, Elastic, and HashiCorp exemplify the approach: each releases a fully functional community edition under an open source license, builds a developer community around it, and then monetizes advanced features — including enterprise authentication, compliance tooling, advanced analytics, and dedicated support — through a proprietary enterprise tier. The strategic logic is compelling: the open source edition functions as a global distribution and marketing channel at near-zero marginal cost, while the enterprise tier captures the willingness to pay of the organizational customers whose risk tolerance and compliance requirements exceed what the community edition provides (Weiss et al., 2022; Shah, 2006). The tension inherent in the open core model is equally characteristic: community contributors may find that the features they most need are progressively migrated to the proprietary tier, and the governance boundary between community and enterprise editions requires constant negotiation between the commercial imperatives of the firm and the expectations of the open source community (O'Mahony, 2007).

6.1.3 SaaS on Open Source: Databricks, Confluent, MongoDB Atlas

A distinct and increasingly dominant commercial model involves building a managed cloud service on top of an open source software product, monetizing operational convenience, reliability, and scalability rather than software features or support contracts. Databricks, built on Apache Spark; Confluent, built on Apache Kafka; and MongoDB Atlas, built on the MongoDB document database, each offer the open source software's functionality as a fully managed cloud service — removing the operational burden of deployment, tuning, patching, and scaling from the customer in exchange for a subscription fee. This model generates revenue from the service layer rather than the software layer, meaning that the source code can remain available for inspection and self-deployment while the commercially attractive managed service remains proprietary (Weiss et al., 2022). It also places these firms in direct structural competition with the major cloud providers, who can replicate the managed service offering using the same open source code — a tension that drove several of the relicensing decisions analyzed in Section 5.2 (Bechara & Lechner, 2024).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

6.1.4 Dual Licensing: MySQL and Qt

The dual licensing model offers the same software simultaneously under two licenses: a copyleft license available for free to users who comply with its terms (including releasing derivative works as open source), and a commercial license available for a fee to users who require the right to incorporate the software into proprietary products without triggering copyleft obligations. MySQL's long-standing dual licensing arrangement under the GPL and a commercial license exemplified this model before Oracle's acquisition; Qt, the cross-platform application framework, similarly offers a dual licensing structure under the LGPL and a commercial license. The model exploits the structural tension between copyleft's compliance requirements and the needs of commercial software developers to create a natural segmentation: developers of open source applications use the free GPL version, while developers of proprietary applications pay for commercial licenses (Weiss et al., 2022; Riehle, 2009). Empirical analysis of dual licensing strategies identifies the commercial licensing component as providing revenue opportunities that correct for the underinvestment problem created by purely open source provision, while the copyleft component ensures that the open source community continues to receive contributions and improvements (Riehle, 2009).

6.1.5 Donations and Foundations: Linux Foundation, Apache, FSF

The foundation model — in which open source development is sustained through donations from corporate sponsors, individual contributors, and institutional grants, administered by a non-profit governance body — represents the oldest and most ideologically consistent open source business arrangement. The Linux Foundation, the Apache Software Foundation, and the Free Software Foundation each provide governance infrastructure, legal services, trademark management, and financial administration for projects that would otherwise lack institutional homes. The Linux Foundation in particular has evolved into a significant industrial consortium: it provides neutral governance for projects whose development requires multi-corporate collaboration without ceding control to any single sponsor — a function that is economically valuable precisely because it resolves the collective action problem that would otherwise prevent competing firms from co-investing in shared infrastructure (Ostrom, 1990; Lerner & Tirole, 2002). However, the foundation model creates its own economic tensions: foundation budgets depend heavily on large corporate sponsors, which creates potential influence dynamics between donor corporations and the ostensibly neutral governance body — a structural conflict of interest that community researchers have identified as a governance challenge requiring institutional design responses (Curto-Millet et al., 2022).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

6.2 Proprietary Software Business Models

6.2.1 Perpetual License: The Classic Model in Decline

The traditional proprietary software business model — in which a customer pays a one-time fee for a perpetual license to use a specific version of a software product — defined the commercial software industry from the late 1960s through the early 2000s. The model's economics were straightforward: the vendor incurred high fixed costs in software development and near-zero marginal costs in distribution, and recouped its investment through per-seat or per-server licensing fees that collectively exceeded development costs over the product lifecycle (Bonaccorsi & Rossi, 2003). The perpetual license model generated powerful lock-in effects: once an organization had integrated a proprietary product into its workflows and trained its workforce on it, the costs of switching to an alternative — including retraining, data migration, workflow redesign, and integration rework — created durable captive customer relationships that reduced competitive pressure on incumbent vendors (Petcu et al., 2013). However, the perpetual license model has declined substantially as a primary revenue mechanism for most large software vendors, largely displaced by subscription-based models that convert one-time license revenue into recurring annuities and provide vendors with more predictable revenue streams and stronger renewal-based retention dynamics (Weiss et al., 2022).

6.2.2 Subscription (SaaS): Microsoft 365, Adobe Creative Cloud, Salesforce

The subscription SaaS model has become the dominant commercial structure for proprietary enterprise and consumer software over the past decade, with Microsoft 365, Adobe Creative Cloud, and Salesforce representing its mature enterprise form. From a vendor economics perspective, the shift from perpetual licensing to subscription improves revenue predictability, reduces the risk of customer defection to older versions, and creates a continuous relationship through which vendors can deliver updates, expand product scope, and identify upsell opportunities (Wen & Choudhury, 2019). From a customer perspective, the subscription model lowers upfront acquisition costs but transforms software from a one-time capital investment into a recurring operational expense that grows with organizational headcount, usage, and the vendor's pricing decisions over time. The dependency dynamics of the SaaS model exceed those of the perpetual license: because SaaS applications are accessed remotely rather than installed locally, customer data and workflows are continuously hosted within the vendor's infrastructure, making migration to alternatives technically complex and operationally disruptive even when the customer's contractual relationship has ended (Alhosban et al., 2024; Petcu et al., 2013).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

6.2.3 Freemium: The Conversion Funnel as Strategy

The freemium model — in which a base version of a proprietary product is made available at no cost as a customer acquisition mechanism, with premium features or usage tiers available on a paid subscription basis — operates as a large-scale behavioral experiment in customer conversion. The free tier functions as a distribution and marketing channel, lowering barriers to trial adoption and enabling viral growth through word-of-mouth and network effects; the premium tier captures revenue from the subset of users whose usage patterns, organizational requirements, or willingness to pay makes conversion economically rational (West & Gallagher, 2006). It is important to distinguish freemium from open source: in the freemium model, the software remains proprietary — the free tier is not a licensing grant but a commercial decision that can be revoked or modified at any time by the vendor (Weiss et al., 2022). Research on the economics of freemium conversion identifies the design of the free/paid boundary as the critical strategic variable: the free tier must provide sufficient value to drive adoption while withholding sufficient value to motivate conversion, a balance that shifts continuously as competitive dynamics and user expectations evolve (Lerner & Tirole, 2002).

6.2.4 Closed Ecosystems: App Stores as Rent-Seeking

The Apple App Store and Google Play Store represent a model of value capture that extends beyond software licensing into platform-mediated market structure. By requiring that all iOS applications be distributed exclusively through the App Store and subjecting them to Apple's review and approval process, Apple creates a structural rent-extraction mechanism: it charges a commission of up to 30% on digital transactions conducted through iOS applications, generating revenue not from software development but from control over distribution infrastructure (Rikap, 2024). This model has been extensively analyzed in the economics of two-sided markets: the platform operator positions itself as a necessary intermediary between application developers and end users, and uses technical and contractual mechanisms to prevent direct relationships that would bypass the platform's commission structure (Lerner & Tirole, 2002). Regulators in multiple jurisdictions have characterized elements of this model as anticompetitive: the European Commission's Digital Markets Act, which entered into force in 2024, explicitly requires Apple to permit alternative app stores and sideloading on iOS in Europe — a direct regulatory intervention in the closed ecosystem architecture that has defined Apple's mobile platform economics (Pohle, 2024; Glasze et al., 2022).

6.3 The Value Paradox in Open Source

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

The most striking illustration of the economic structure of open source software is the extraordinary disparity between its supply-side cost and its demand-side value. A 2024 study by Hoffmann, Nagle, and Zhou at Harvard Business School, drawing on unique global usage data from over nine million company websites and from the Linux Foundation's Census II of Open Source Software, provides the most rigorous quantitative estimate to date of both dimensions. The supply-side value of the most widely used open source software — the cost to recreate it once — is estimated at $4.15 billion (based on a 2024 Harvard Business School study analyzing usage data from over nine million company websites; Hoffmann et al., 2024), a substantial but comprehensible figure. The demand-side value — the aggregate replacement cost that all firms using open source software would face if they had to build it internally — is estimated at $8.8 trillion, more than 2,000 times the supply-side cost, and 3.5 times current global software expenditure (Hoffmann et al., 2024). The study further finds that the value generated is highly concentrated: the top six programming languages account for 84% of total demand-side value, and approximately 5% of developers generate approximately 95% of the supply-side value — a distribution consistent with the power law dynamics observed in other public goods production systems (Hoffmann et al., 2024; Lerner & Tirole, 2002).

This 2,000-fold value differential between production cost and aggregate social value constitutes the defining economic problem of open source software. It means that society as a whole extracts vastly more value from open source than it invests in its creation and maintenance, generating a structural underinvestment dynamic in which the infrastructure that sustains the entire digital economy is chronically underfunded relative to its economic significance (Hoffmann et al., 2024; Walden, 2020). The free rider problem identified in Section 3.4 operates not merely at the level of individual developers but at the level of entire industries: research on open source contribution patterns consistently finds that the overwhelming majority of firms that depend heavily on specific open source projects contribute little or nothing to their maintenance (Lerner & Tirole, 2002; Dahlander & Magnusson, 2008). The structural consequence is a system in which critical digital infrastructure is often maintained by small numbers of underfunded volunteer developers — a fragility memorably illustrated by Randall Munroe's widely reproduced "Dependency" cartoon (xkcd no. 2347), which depicts the entire modern digital stack resting on a single precariously balanced structure maintained by one person in Nebraska — a visual metaphor that the XZ Utils backdoor episode of 2024 rendered almost literal (Pirozzi et al., 2025; OpenSSF, 2024).

6.4 Total Cost of Ownership (TCO)

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

The economic comparison between open source and proprietary software at the organizational level is most rigorously framed through the concept of total cost of ownership, which encompasses all costs associated with acquiring, deploying, operating, maintaining, and eventually replacing a software system over its full lifecycle — not merely the upfront licensing or acquisition cost (Shaikh & Cornford, 2011). Open source software is frequently described as "free," but this characterization reflects only its zero acquisition price and systematically understates its actual organizational cost. Empirical TCO studies consistently identify significant hidden costs associated with open source adoption: the cost of internal expertise to deploy and maintain software without vendor support; the cost of security auditing and license compliance management; the cost of integration with existing enterprise systems that may have been designed around proprietary alternatives; and the cost of organizational change management for users transitioning from familiar proprietary interfaces (Kodhek, 2024; Shaikh & Cornford, 2011). Research on open source TCO is cheaper in terms of direct costs — hardware, software, and support — but that it is difficult to measure indirect costs since these are hidden and may vary within the business and software environment (Dewi, 2009).

The TCO profile of proprietary software presents a complementary set of advantages and risks. Proprietary vendors typically include support, security patching, and version upgrade paths in their licensing fees, providing cost predictability that open source deployments — dependent on internal expertise or third-party service providers — cannot guarantee without additional contracting (Bonaccorsi & Rossi, 2003). However, the predictability of proprietary TCO is bounded by the vendor's pricing decisions, which change over the lifecycle of the customer relationship: enterprises that have built deep dependencies on proprietary systems frequently find that licensing costs escalate at renewal, that features previously included in base licenses migrate to premium tiers, and that the switching costs that make migration prohibitively expensive give the vendor effective pricing power that erodes the initial cost justification for the proprietary choice (Petcu et al., 2013; Alhosban et al., 2024). A UK Cabinet Office report on the TCO of open source software, one of the most comprehensive empirical studies of comparative TCO in the public sector context, found that over half of respondents had replaced proprietary software with open source alternatives and reported net savings, while emphasizing that successful open source adoption requires strategic planning, genuine organizational commitment, and realistic assessment of internal capabilities — and should not be approached as an easy cost-reduction exercise (Shaikh & Cornford, 2011).

6.5 Investment and Market Consolidation

6.5.1 Acquisitions and the Commercialization of Open Source Communities

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

The acquisition of major open source projects and companies by large technology corporations has become one of the defining structural dynamics of the contemporary software industry, raising fundamental questions about whether commercial consolidation of open source assets can be reconciled with the governance principles and community norms that make those assets valuable. IBM's $34 billion acquisition of Red Hat in 2019, Microsoft's $7.5 billion acquisition of GitHub in 2018, and Salesforce's $27.7 billion acquisition of Slack represent a pattern in which the most commercially valuable open source-adjacent assets are progressively absorbed into the portfolios of large proprietary technology corporations (IBM, 2019; Rikap, 2024). The theoretical risk is that open source projects developed within large corporate owners shift governance priorities from community responsiveness to revenue maximization — a concern that has been empirically validated in several cases, most notably MySQL under Oracle's stewardship (O'Mahony, 2007).

6.5.2 Venture Capital and the Commercialization Tension

The proliferation of venture capital investment in open source companies since the mid-2010s has introduced a structural tension between the governance expectations of open source communities and the financial return requirements of VC-backed firms. Digital startups that engage with open source communities during their conception and commercialization stages benefit more from inbound open source contributions, whereas those in growth stages benefit more from outbound contributions — a pattern consistent with firms using open source community engagement strategically, shifting from consumers to contributors as their commercial trajectory develops (Nagle et al., 2024). However, the VC exit orientation — which typically requires a liquidity event through acquisition or IPO within a defined time horizon — creates structural incentives for portfolio companies to prioritize revenue growth and profitability over community governance commitments that do not directly convert to financial returns (Cooiman, 2024). The relicensing decisions of HashiCorp, Elastic, and MongoDB — each venture-backed companies that transitioned from OSI-approved licenses to more restrictive terms following periods of rapid commercial scaling — are consistent with this structural dynamic: the community-building benefits of open source are most valuable during the growth phase, while the revenue-protection benefits of restrictive licensing become more attractive as commercial scale is achieved (Bechara & Lechner, 2024; West & Gallagher, 2006).

6.5.3 Platform Decay and the "Enshittification" Phenomenon

A recently theorized dynamic that is directly relevant to the open source versus proprietary debate is the phenomenon of platform decay, described by writer and activist Cory Doctorow through the neologism "enshittification," which the American

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Dialect Society selected as its 2023 Word of the Year, and which has since been formally adopted as an analytical concept in organizational and labor research. Doctorow's framework proposes a three-stage model of platform decline: platforms initially offer genuine value to users in order to establish adoption and network effects; they then degrade the user experience to extract value for business customers; and finally they degrade both user and business customer experience to maximize returns to shareholders — a process enabled by the combination of switching costs, network effects, and reduced competitive discipline in concentrated platform markets (Doctorow, 2023). Academic researchers have extended Doctorow's framework beyond consumer platforms to analyze institutional dynamics in academic publishing and gig-economy labor relations, finding that the same structural mechanisms — captive user bases, exit barriers, and monopoly pricing power — drive quality degradation across a wide range of platform-mediated markets (Linnenluecke et al., 2025; Maffie & Hurtado, 2025). Applied to the open source versus proprietary debate, enshittification offers an important analytical lens: projects that begin as genuinely open, community-governed commons can undergo a structural transformation — through VC investment, corporate acquisition, or relicensing — that progressively extracts value from the community that created the project while reducing the openness and governance participation that gave the community its motivation to contribute in the first place (Doctorow, 2023; O'Mahony, 2007; Bechara & Lechner, 2024).

7. Organizational and Governance Dimensions

7.1 Governance of Open Source Projects

7.1.1 Governance Models: BDFL, Committee, and Foundation

Open source projects do not self-organize spontaneously; they require explicit governance structures that determine who holds decision-making authority, how conflicts are resolved, how new contributors gain recognition, and how the project sustains itself over time. Research analyzing 101 software foundations and the projects they govern identifies a typology of governance models that span a spectrum from highly centralized to broadly distributed decision-making (Cosentino et al., 2020). The Benevolent Dictator for Life (BDFL) model places final decision-making authority with a single founding leader — typically the original author of the project — who exercises veto power over all major technical and governance decisions. Python under Guido van Rossum, Linux under Linus Torvalds, and Drupal under Dries Buytaert are canonical examples. The BDFL model's advantages are architectural

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

coherence, decisive leadership, and alignment around a founding vision; its principal risks are the bottleneck effect as the project scales, the entrenchment of a single individual's values and biases, and catastrophic fragility if the BDFL becomes unavailable (Cosentino et al., 2020; O'Mahony, 2007).

Committee-based governance models distribute decision-making across a body of recognized contributors, typically elected or appointed based on meritocratic criteria of contribution history. The Apache Software Foundation pioneered the meritocratic governance model, in which committees have final decision-making authority over project content and direction, overseen by the foundation's board, with decisions typically made by lazy consensus: a few positive votes with no negative vote constitutes approval, and negative votes must include technical justification (Cosentino et al., 2020). Foundation-backed governance, in which a non-profit or trade association provides the institutional framework for a project's legal, financial, and brand stewardship, enables a degree of neutrality from any single corporate sponsor and provides mechanisms for managing intellectual property contributions — functions that are increasingly important as open source projects accumulate economic and strategic value (Cosentino et al., 2020; Ostrom, 1990). Many mature projects combine governance models: a project hosted within the Linux Foundation or Apache Software Foundation may also self-govern as a do-ocracy or maintain a technical steering committee with BDFL-like characteristics at the level of specific subsystems (Cosentino et al., 2020).

7.1.2 The Succession Problem

A structural vulnerability common to all open source governance models is the succession problem: the difficulty of maintaining project continuity when key contributors — particularly founding leaders or sole maintainers — become unavailable through burnout, career change, illness, or death. The BDFL model concentrates this risk acutely: when Guido van Rossum abdicated from his BDFL role in Python in 2018, citing burnout as the primary reason, the community was required to rapidly design and adopt a new governance structure under conditions of significant uncertainty (Curto-Millet et al., 2022). Research on the sustainability of open source commons identifies maintainer continuity as one of the most critical and most poorly addressed dimensions of project sustainability, finding that the relationships among resource, infrastructure, and interactional sustainability dimensions are frequently characterized by tensions that leave projects structurally unprepared for key-contributor departure (Curto-Millet et al., 2022). The problem is particularly acute for the long tail of smaller, widely used open source projects — including critical security libraries and utility components — that are effectively maintained by a single volunteer, as demonstrated by the XZ Utils backdoor case analyzed in Section 5.7,

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

in which the sole maintainer's state of exhaustion and isolation created the opening that a sophisticated attacker exploited over a period of years (Pirozzi et al., 2025).

7.1.3 Diversity and Inclusion: The Representation Crisis

The open source developer community exhibits demographic characteristics that are far more homogeneous than either the professional software industry or the general population. Survey data consistently show that men constitute approximately 91–95% of open source contributors, a gender disparity that exceeds even the already significant gender gap in the broader computing industry (Trinkenreich et al., 2021). Research analyzing the perceptible ethnic and gender diversity of DevOps and non-DevOps open source contributors found that contributors perceptible as non-White — including Hispanic and Black contributors — are greatly underrepresented, and that DevOps contributors are significantly less diverse than non-DevOps contributors (Izaiku et al., 2023). Critically, this underrepresentation is not merely a passive reflection of pipeline demographics but is actively reproduced by project norms and culture: research on gender bias in GitHub found that code written by women was accepted more often (78.6%) than code written by men (74.6%) when gender was not identifiable, but that women's code acceptance rates dropped significantly when gender was immediately apparent from username or profile — a pattern consistent with implicit gender bias operating at the point of contribution review (Terrell et al., 2017). Heightened levels of stress brought on by unhealthy interactions may make it harder for projects to attract, include, and retain a diverse talent pool, suggesting that inclusion and sustainability are structurally linked rather than independent governance concerns (Raman et al., 2020).

7.1.4 Maintainer Burnout: The Silent Crisis of Digital Infrastructure

Maintainer burnout has emerged as one of the most significant and most understudied governance challenges in the open source ecosystem. Survey data from Intel's 2023 open source contributor survey identified maintainer burnout as the top challenge among respondents, reported by 45% of those surveyed — ahead of documentation, sustainability, and community engagement concerns (Degefa et al., 2024). The structural causes of burnout in open source are well documented: key stressors include high expectations regarding workload, the predominantly volunteer nature of contributions, social isolation, and a lack of formal support structures; additionally, research highlights a significant lack of awareness of available intervention strategies, further increasing the risk of burnout and stress among community members (Degefa et al., 2024). Research on toxic interactions in open source communities found that developers frequently report GitHub notifications as a constant stream of negativity, and that toxic conversations systematically demotivate

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

and burn out developers, creating challenges for sustaining open source projects over time (Raman et al., 2020). The World Health Organization's official recognition of burnout as an occupational phenomenon — defined as a syndrome resulting from chronic workplace stress that remains unmanaged, characterized by exhaustion, cynicism, and inefficacy — provides the conceptual framework for understanding maintainer burnout as a genuine occupational health issue rather than a personal failing, even in the voluntary context of open source development (Demerouti, 2024). The causes and remedies of maintainer burnout are themselves debated: Eghbal (2020) frames the problem primarily as one of funding and economic incentives, while Geiger et al. (2021) emphasize community norms and interaction quality as equally determinative. A third perspective, advanced by Trinkenreich et al. (2023), identifies intersectional factors — gender, geography, and employment status — as structural predictors of burnout risk that funding alone does not address. These perspectives suggest that no single intervention is sufficient and that the governance response must be multi-dimensional.

The systemic consequences of maintainer burnout extend well beyond the individual. When the team — or, shockingly often, single person — maintaining a project is running on fumes, exploits get missed and malicious collaborators go undetected, putting the security of a significant proportion of global software infrastructure at risk (Degefa et al., 2024). The OpenSSF's Alpha-Omega project, established precisely to provide sustained funding for security audits and maintainer support in critical open source infrastructure, represents an institutional acknowledgment that burnout is not an individual problem but a structural failure of the governance and economic arrangements surrounding open source software (OpenSSF, 2024; Walden, 2020).

7.2 Corporate Adoption of Open Source

7.2.1 InnerSource: Applying Open Source Principles Inside Organizations

InnerSource — a term coined by Tim O'Reilly in 2000 and formally conceptualized by Stol and Fitzgerald (2015) — refers to the practice of applying open source development principles within the boundaries of a single organization to software that remains proprietary. Rather than treating internal codebases as siloed departmental assets, InnerSource initiatives make code, documentation, and development processes transparent and accessible to all employees, and adopt open source-style contribution mechanisms — pull requests, issue trackers, committer hierarchies — to enable cross-team collaboration (Stol & Fitzgerald, 2015). Empirical research on InnerSource adoption finds that the practice helps create social capital by emphasizing the importance of relationships between developers and teams interacting across intra-organizational boundaries, which in turn links to higher levels

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

of job satisfaction and reduced information duplication (Stol & Fitzgerald, 2015). The Bosch Group, one of the most extensively documented InnerSource implementations in the literature, launched its InnerSource initiative in 2009; by 2024 the InnerSource community at Bosch had grown to approximately 34,000 members from over 150 business units across more than 50 countries, with approximately 12,000 InnerSource repositories and one million code contributions, demonstrating the model's scalability in large industrial organizations.

7.2.2 Open Source as Competitive Strategy: Commoditizing the Competitor's Stack

Major technology corporations use open source contributions not only as engineering tools but as strategic competitive instruments. The logic — identified in academic analysis of corporate open source behavior — is that by commoditizing components at the layers of the stack where competitors derive proprietary advantage, a firm can erode those advantages while retaining its own differentiation at the service layers where it captures commercial value (West & Gallagher, 2006). Google's donation of the Android operating system to the Open Handset Alliance effectively prevented any single competitor from establishing a proprietary mobile operating system monopoly, while retaining Google's competitive position at the application and services layer through Google Mobile Services (Rikap, 2024). Meta's open-sourcing of the PyTorch framework and React front-end library similarly commoditized AI infrastructure and web development tooling, reducing barriers to adoption of technologies that funnel developers into Meta's broader ecosystem while eliminating the possibility of proprietary differentiation at those layers by competitors (Rikap, 2024; West & Gallagher, 2006). This behavior is consistent with the theoretical prediction that platform-owning firms strategically open the layers of the stack that complement their proprietary differentiators while keeping the value-capturing layers closed — a dynamic that shapes not only what gets open-sourced but how open source governance evolves over time (West & Gallagher, 2006).

7.2.3 Corporate Contribution Policies and the Capture of Neutral Projects

The participation of large technology corporations in ostensibly neutral open source foundations has generated concerns about the effective capture of community governance by corporate interests. Research on the funding and governance models of open source projects finds that corporate-backed projects and foundation-backed projects exhibit significantly different development activity patterns: foundation-backed projects with diversified funding bases maintain more consistent community governance, while projects with dominant single-sponsor relationships are more susceptible to governance shifts when that sponsor's commercial priorities change

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

(Curto-Millet et al., 2022; Nagle et al., 2024). The concentration of Linux kernel contributions in a small number of large technology corporations — consistently, over half of all kernel commits originate from employees of the top ten corporate contributors, including Intel, Red Hat/IBM, Google, and Samsung — illustrates how community governance and corporate development priorities can converge in ways that serve the dominant contributors' architectures and hardware platforms (Dahlander & Magnusson, 2008). Digital startups that engage with open source communities during their conception and commercialization stages benefit more from inbound open source contributions, whereas those in growth stages benefit more from outbound contributions — a pattern consistent with firms using open source community engagement strategically as their commercial trajectory develops (Nagle et al., 2024).

7.2.4 License Compliance: The Legal Nightmare Ignored by Most Organizations

Open source license compliance has evolved from a niche concern of legal departments into a mainstream enterprise risk management issue, driven by the ubiquity of open source components in commercial software and the increasing willingness of courts to enforce license terms in litigation. The 2024 Open Source Security and Risk Analysis (OSSRA) report, produced by Synopsys (Black Duck) from audits of commercial software codebases, found that over 53% of audited codebases contained open source components with license conflicts — a proportion that illustrates the scale of the compliance challenge facing the industry (Synopsys, 2024). Research on the certification of open source license compliance finds that the OSI/IEC 5230:2020 OpenChain standard — which specifies a documented OSS policy, clearly defined roles and responsibilities, and the ability to provide a software bill of materials — is held by companies including GE Digital, Google, Microsoft, Siemens, Sony, and Toshiba, but that third-party certification at the required level of rigor involves significant costs from auditor agency fees, creating structural compliance disparities between large enterprises and smaller organizations that cannot absorb these costs (Zaggl et al., 2025). Two landmark 2024 court cases — Sebastian Steck v. AVM, in which an individual developer successfully enforced GPL license obligations against a corporate electronics manufacturer, and Entr'ouvert v. Orange S.A., in which the Paris Court of Appeal ordered a telecom operator to pay over €900,000 in damages for GPL non-compliance — have demonstrated that open source license enforcement is not theoretical but actively pursued, and that the financial consequences of non-compliance are material (Gangadharan et al., 2009; Zaggl et al., 2025).

7.3 Governance of Proprietary Software

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

7.3.1 Closed Roadmap: The Customer as Stakeholder Without a Voice

A fundamental governance asymmetry of proprietary software is that customers have no formal mechanism to influence the product roadmap beyond the commercial signals of their purchasing decisions. Unlike open source projects — in which any stakeholder can propose changes, file issues, submit patches, and participate in governance processes — proprietary software development proceeds according to the vendor's internal priorities, competitive positioning, and business model logic, with customer feedback incorporated at the vendor's discretion (Fitzgerald, 2006). This asymmetry becomes structurally significant in enterprise software markets where switching costs are high enough to render the exit option impractical: a customer deeply dependent on a proprietary ERP system, database, or cloud platform cannot easily exit in response to product decisions it dislikes, and must therefore accept feature priorities, deprecation timelines, and pricing changes that serve the vendor's interests rather than its own (Petcu et al., 2013; Alhosban et al., 2024). The consequences can be severe: research on enterprise software procurement identifies a pattern in which vendor roadmap decisions — feature migration to premium tiers, deprecation of compatibility interfaces, enforcement of new licensing models — impose significant unplanned costs on customer organizations that lack contractual protections against such changes (Jansen et al., 2009).

7.3.2 Product Discontinuation: When the Vendor Decides to Stop

Software product discontinuation — or sunsetting — is the process by which a vendor ends active development, maintenance, and support for a product that remains in use by customers. Research on the software product lifecycle identifies sunsetting as a "blind spot" in software product management literature: despite its significant operational impact on customers, the practice lacks standardized industry frameworks comparable to those governing other phases of the product lifecycle (Jansen et al., 2009). The customer impact of sunsetting is multidimensional: customers lose access to security patches, leaving deployed systems permanently vulnerable to subsequently disclosed vulnerabilities; they face compatibility degradation as the sunsetted software fails to integrate with updated adjacent systems; and they must absorb the transition costs of migrating to replacement systems that may require substantial retraining, data migration, and integration rework (Jansen et al., 2009). Contemporary examples illustrate the organizational disruption involved: Broadcom's acquisition of VMware resulted in the accelerated retirement of support for the widely deployed vSphere 7 platform; SAP's end-of-support date of 2027 for its legacy ECC ERP platform has been estimated to affect approximately 17,000 organizations that will still be running the legacy system at that time. These cases demonstrate that sunsetting risk is not an edge case but a

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

structural feature of reliance on proprietary software in enterprise contexts, and that its organizational impact significantly exceeds what is captured in initial TCO calculations (Jansen et al., 2009; Petcu et al., 2013).

7.3.3 Mergers and Acquisitions: The Risk of the Sunsetted Acquisition

Mergers and acquisitions involving software companies create a specific category of customer risk: the acquired product may be discontinued, restructured, or repriced as the acquiring entity rationalizes its product portfolio or eliminates competition with its own offerings. Research on enterprise software acquisition identifies the misalignment between the acquiring firm's strategic objectives and the acquired product's customer commitments as a principal source of post-acquisition value destruction for enterprise customers (Wen & Choudhury, 2019). When an organization builds critical processes on a proprietary product that is subsequently acquired, the acquiring entity may change licensing terms, discontinue the product, discontinue support for integrations with competing products, or accelerate the migration timeline in ways that impose costs the customer did not anticipate at the time of original procurement (Jansen et al., 2009). The MySQL/Oracle case, the Broadcom/VMware case, and the IBM/Red Hat case each illustrate different dimensions of this risk: Oracle's acquisition of MySQL raised community concerns about stewardship that led to a significant fork; Broadcom's acquisition of VMware led to controversial changes in licensing and support policies for a product embedded in a large proportion of enterprise data center infrastructure; and IBM's acquisition of Red Hat, while generally regarded as preserving the open source governance of the acquired projects, represented a concentration of the most commercially successful open source business model within a large proprietary technology conglomerate (O'Mahony, 2007; Rikap, 2024). From the customer's perspective, the governance risk of proprietary software is ultimately the governance risk of depending on a vendor whose organizational priorities may diverge at any time from the commitments that originally justified the procurement decision — a risk that has no structural equivalent in community-governed open source software, where the right to fork provides a last-resort protection against governance capture.

8. Social, Cultural, and Political Impact

8.1 Democratization of Access to Technology

The relationship between open source software and technological democratization in developing countries is a genuinely contested empirical question — one whose

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

optimistic assumptions have been substantially qualified by evidence since the mid-2000s. The theoretical case for open source as a vector of digital inclusion rests on two propositions: that the elimination of licensing costs removes a significant barrier to software adoption in low-income contexts, and that transparency of code enables local customization, localization, and capacity building in ways that proprietary software does not (Biström et al., 2024; Bechara & Lechner, 2024). These propositions are not unfounded, but they are more conditional in practice than their advocates have sometimes acknowledged.

The One Laptop per Child (OLPC) initiative, launched in 2005 with the goal of providing low-cost, open source-equipped laptops to children in the developing world, represents the most ambitious and most thoroughly evaluated test case of the democratizing potential of open source educational technology. The XO laptop ran a customized Linux distribution with the Sugar graphical interface — itself an open source project designed around constructionist learning principles — and was deployed across approximately 40 countries, with Uruguay's Plan Ceibal becoming the first national deployment to provide every primary school child with a personal device by 2009 (Cristia et al., 2017). However, a large-scale randomized evaluation of the OLPC program in 531 rural primary schools in Peru, with a ten-year follow-up, found no significant effects on academic performance, grade progression from primary to secondary school, or university enrollment (Cristia et al., 2017). These null results are consistent with a broader literature on ICT for Development showing that technology access without adequate teacher training, pedagogical integration, and complementary institutional support generates limited and often negligible educational outcomes — a finding that applies regardless of whether the software is open source or proprietary (Warschauer & Ames, 2010).

The broader digital divide literature frames this limitation in structural terms. The digital divide is not merely a binary access divide — defined by whether one has or lacks internet connectivity or computing devices — but a multi-layered phenomenon encompassing skills, usage quality, and the social structures that shape whether digital access translates into substantive opportunity (Ragnedda & Muschert, 2013; van Dijk, 2020). Open source software addresses at most one dimension of this structure — the cost barrier — while leaving the skills deficit, infrastructure inadequacy, and pedagogical gap substantially unchanged. The invisible barrier of open source is, accordingly, not cost but complexity: open source software frequently demands greater technical literacy to install, configure, troubleshoot, and maintain than proprietary alternatives that invest in user interface polish and user support infrastructure (Ven & Verelst, 2008). The democratizing promise of open source in low-income and low-connectivity contexts is therefore real but severely conditional on

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

investments in human capital, institutional support, and pedagogical integration that open source licensing alone cannot provide.

8.2 Digital Sovereignty and National Security

The national security dimension of the open source versus proprietary debate has been elevated to the highest levels of government policy by a succession of incidents that revealed how proprietary software — particularly foreign-developed security software with privileged system access — can function as a vector of state-level intelligence exploitation. The foundational mechanism is structural: proprietary security software, by definition, cannot be independently audited by the governments or organizations that deploy it, and simultaneously possesses deep access privileges to the systems it protects — making it a plausible high-value target for foreign intelligence agencies that can compel or coerce the software vendor — a governance risk assessment supported by documented policy responses rather than confirmed intelligence operations (Pohle, 2024; Glasze et al., 2022).

The Kaspersky Lab case constitutes the paradigmatic contemporary illustration of this structural risk. The U.S. Department of Homeland Security's 2017 directive ordering the removal of Kaspersky-branded products from all federal information systems, subsequently codified in the National Defense Authorization Act for Fiscal Year 2018, was grounded in the assessment that Kaspersky software's privileged access to customer systems and the Russian government's legal authority to compel Russian companies to cooperate with intelligence agencies created an unacceptable national security exposure (U.S. Department of Commerce, 2024). The U.S. Bureau of Industry and Security's June 2024 Final Determination — the first of its kind under the Information and Communications Technology and Services Supply Chain authorities — formally prohibited Kaspersky from providing antivirus and cybersecurity products or services to U.S. persons, citing a determination that Kaspersky's continued operations posed undue and unacceptable national security risk, including the potential to identify sensitive U.S. person data, transfer it to Russian government actors, and install malicious software on customers' systems (U.S. Department of Commerce, 2024). Analogous concerns have shaped European policy toward Huawei's 5G infrastructure equipment — a case where the hardware-software integration of proprietary telecommunications systems created similar structural concerns about the ability of the Chinese state to exploit vendor access for intelligence purposes. Germany, despite its economic ties to China and status as Europe's largest telecom market, only reached a tentative agreement on Huawei restrictions in July 2024, reflecting the deep tension between digital sovereignty aspirations and economic interdependency (Institut français des relations internationales, 2025).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

National open source software policies have emerged across multiple jurisdictions as a structural response to the dependency vulnerabilities created by reliance on foreign proprietary software. Germany's Online Access Act 2.0, agreed in February 2024, explicitly mandates that government entities prefer open source software over licensed alternatives that restrict use, distribution, transfer, or modification — the first binding German legislation to enshrine an open source preference in public procurement (Neudert, 2024). France's policy on open source in public administration, formalized through the Socle Interministériel de Logiciels Libres, maintains a list of recommended open source software for government use and has been a model for similar initiatives in European countries. India's approach to digital sovereignty through open source has been institutionalized through its Digital Public Infrastructure stack — the Aadhaar digital identity system, the UPI payments infrastructure, and the ONDC open commerce network — each built on open source or open standards foundations as a means of enabling population-scale deployment without dependence on foreign proprietary platforms (Srivastava & Mishra, 2025). Brazil's SERPRO (Federal Data Processing Service), established in 1964 and currently employing over 10,000 technical staff, has been a vehicle for the development and deployment of open source solutions in Brazilian federal government operations, with the Brazilian open government data trajectory beginning with the e-government program launched twenty-five years ago and generating successive layers of open data policy and infrastructure (Santos et al., 2024).

8.3 Education and Talent Development

Open source development has long been theorized as a form of informal apprenticeship — a publicly documented, peer-reviewed, cumulative learning environment that offers technical education of a character that structured academic curricula struggle to replicate (Lakhani & von Hippel, 2003). The act of contributing to an open source project requires a developer to engage with a real, production-quality codebase; to understand code written by others across multiple architectural layers; to submit code changes for peer review by experienced practitioners; and to respond to technical feedback in a public forum that creates durable records of both errors and improvements. These characteristics constitute a form of learning that is simultaneously more authentic and more immediately applicable than most university programming assignments, and the public visibility of contribution history on platforms such as GitHub creates a portfolio of verified, peer-reviewed work that can function as evidence of technical capability in employment contexts (Trinkenreich et al., 2021).

However, the notion that open source contribution history on GitHub constitutes a meritocratic portfolio — one that allows talent to be recognized independently of credentials and institutional affiliation — has been substantially complicated by

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

empirical research. Open source software is commonly described as a meritocracy where decisions are based solely on technical merit, but the literature identifies a complex social structure underlying the apparent meritocracy: research on pull request acceptance in GitHub found that both technical quality and social connection between the submitter and the project manager significantly influence contribution acceptance, meaning that who you know mediates what you can contribute (Tsay et al., 2014). As analyzed in Section 7.1.C, gender bias further undermines the meritocratic ideal: women's contributions are evaluated differently when their gender is identifiable, suggesting that the allegedly neutral contribution review process reproduces social biases (Terrell et al., 2017). The GitHub portfolio model also disadvantages developers from low-resource contexts — those without reliable high-speed internet access, those with caregiving responsibilities that limit available time for unpaid open source contribution, and those whose employment contracts restrict participation in external projects — creating a credential system that structurally privileges those already advantaged by geography, gender, and economic circumstance (Trinkenreich et al., 2021).

At the institutional level, the relationship between open source and research universities has been shaped by the movement for open science and reproducible research. The reproducibility crisis in computational science — in which significant proportions of published computational results cannot be reproduced from the published methodology alone — has been directly linked to the prevalence of proprietary software tools in scientific workflows, whose outputs cannot be independently verified without access to the proprietary code (Stodden et al., 2016). The open source movement in science — embodied in tools such as Python, R, Jupyter, and the broader scientific Python ecosystem — has enabled a model of computational research in which code, data, and results are published together as an integrated, reproducible artifact, significantly improving the verifiability and cumulative buildability of scientific knowledge (Stodden et al., 2016; Hoffmann et al., 2024).

8.4 Diversity, Equity, and Inclusion in the Tech Ecosystem

The demographic homogeneity of open source contributor communities — documented in Section 7.1.C — represents not merely a governance challenge for individual projects but a broader equity issue with implications for what technologies get built, whose needs they address, and how benefits from digital innovation are distributed across society. A systematic review of women's participation in open source finds that women remain significantly underrepresented as contributors despite comprising a growing proportion of the software industry workforce, and that the structural barriers are attributable to a combination of unwelcoming community culture, implicit gender bias in contribution review, a lack of visible role models and

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

mentorship, and the particular hostility of some project communities to newcomers who do not conform to established norms (Trinkenreich et al., 2021).

Structured inclusion initiatives have emerged as targeted responses to these structural barriers. Outreachy, which runs paid remote internships for people subject to systemic bias in technology, has placed participants from more than 60 countries in open source contributions since its founding in 2006. Google Summer of Code, which provides stipends for student contributors to open source organizations during summer periods, has funded over 20,000 student participants since 2005, with an explicit mandate to expand participation beyond the demographics that dominate spontaneous open source contribution. She Code Africa, founded in 2019 and operating across more than 50 African countries, combines open source skill development with community building and mentorship explicitly targeted at women in the African technology ecosystem. These initiatives represent an acknowledgment by the open source community that the removal of legal barriers to participation — the opening of source code and governance processes — is insufficient to produce genuine inclusion in the absence of active structural interventions (Trinkenreich et al., 2021; Raman et al., 2020).

The proprietary software model's relationship to diversity and inclusion presents a distinct but equally significant set of concerns. While large proprietary technology corporations have invested significantly in diversity and inclusion programs, the certification and training ecosystems that define career entry pathways in the enterprise software market — including Salesforce certifications, Oracle database certifications, and Microsoft Azure credentials — create barriers to entry that reinforce economic and geographic inequalities. Unlike open source software, whose acquisition and learning costs are in principle zero, proprietary vendor certification tracks require substantial financial investment in training materials, examination fees, and software licenses, effectively excluding aspirants from low-income contexts who cannot absorb these costs without employer sponsorship (Ven & Verelst, 2008; Bonaccorsi & Rossi, 2003).

8.5 Environmental Impact

The environmental dimensions of the open source versus proprietary software debate have received substantially less academic attention than their governance, economic, and security counterparts, but are increasingly salient given the growing recognition that software design choices have measurable and significant energy and carbon implications at scale. The ICT sector accounts for an estimated 3–5% of global greenhouse gas emissions, a share that is growing as cloud computing, AI inference, and IoT deployments expand (Freitag et al., 2021). Within this sector, software design

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

choices — including architectural patterns, code quality, algorithmic efficiency, and deployment configuration — have a material effect on energy consumption that is largely independent of hardware selection (Connolly et al., 2025; Georgiou et al., 2022).

The relationship between open source and energy efficiency is theoretically favorable but empirically under-documented. The auditability of open source code in principle enables independent measurement, optimization, and verification of energy efficiency in a manner that proprietary software does not allow — since the energy consumption of computations performed on proprietary closed-source systems cannot be evaluated by the systems' users or regulators (Chowdhury et al., 2025). Research on AI model energy measurement illustrates this asymmetry concretely: open-weight AI models can be measured for per-query energy consumption using hardware performance counters and open source measurement tools; proprietary AI models accessed via API — including GPT-4, Claude, and Gemini — cannot be measured by users because their inference infrastructure is entirely opaque (Chowdhury et al., 2025). The inability to measure the energy cost of proprietary AI inference constitutes a genuine transparency deficit with implications for organizational sustainability reporting and regulatory compliance with emerging carbon disclosure requirements.

Open source software's structural auditability also enables the documentation and remediation of energy inefficiencies — so-called software bloat — that accumulate in codebases over time as features are added without corresponding attention to computational efficiency. A systematic review of the impact of software design elements on energy performance finds that code design choices — including design patterns, refactoring strategies, and code structure — have measurable and sometimes substantial effects on energy consumption, with the magnitude varying significantly by programming language, architectural pattern, and deployment context (Connolly et al., 2025). The community governance model of open source, in which energy efficiency improvements can be proposed, reviewed, and merged by any contributor with the motivation and skill to implement them, offers a structural mechanism for cumulative energy optimization that proprietary development — in which optimization priorities are determined by the vendor's commercial calculus rather than a community of users — may not replicate as effectively (Georgiou et al., 2022).

The Right to Repair movement represents a third environmental dimension where the open source versus proprietary distinction becomes directly relevant. Proprietary embedded software — including the firmware in consumer electronics, agricultural equipment, and medical devices — has been identified as a mechanism of planned obsolescence that forces device replacement rather than enabling repair and

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

extension of product lifetime (Mikolajczak & Trogal, 2024). The EU Right to Repair Regulation, adopted in 2024 and expanding on the 2019 EcoDesign Implementing Regulations, requires manufacturers of certain product categories to provide access to spare parts and technical information to professional repairers — but contains significant limitations in that it does not address the intellectual property restrictions that prevent the modification or redistribution of proprietary embedded software needed for repair (Mikolajczak & Goanta, 2023). Academic analysis of the right to repair in EU law argues that intellectual property law — and particularly copyright, which attaches automatically to software — must play a more central role in right to repair policy, and specifically that the reproduction and modification of proprietary embedded software for repair purposes requires an explicit user rights framework within copyright law rather than merely procurement or product design mandates (Mikolajczak & Goanta, 2023). Open source firmware and embedded software, by contrast, grants the user the legal right to modify, repair, and redistribute the software — making open source licensing a structural enabler of device longevity and repairability that proprietary licensing structurally restricts in most documented deployment contexts, though systematic comparative evidence on actual repair rates across the two models remains limited.

9. Paradigmatic Case Studies

9.1 Linux vs. Windows Server: The War That Open Source Won

The competitive history between Linux and Windows Server constitutes the foundational empirical case study of the open source versus proprietary debate — a longitudinal natural experiment whose outcome has been sufficiently definitive to serve as the benchmark against which all subsequent domain-specific comparisons are measured. As established in Section 4.4, Linux's ascent to dominance in the server and cloud infrastructure market did not occur through a single decisive confrontation but through a process of cumulative advantage that unfolded across two decades of architectural maturation, community growth, and strategic corporate adoption (Fitzgerald, 2006; West & Gallagher, 2006).

The inflection point in the Linux-Windows server competition is traceable to the intersection of two structural developments in the mid-2000s: the explosive growth of web-scale internet services, which required infrastructure economics radically different from those of the enterprise data center market that Microsoft's licensing model had been designed to serve, and the emergence of the LAMP stack as a proven, production-grade architecture capable of supporting those requirements at

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

near-zero licensing cost (Pfandzelter & Bermbach, 2024). Google, Amazon, and the emerging generation of Web 2.0 firms built their infrastructure on Linux not because of ideological preference but because the combination of zero licensing cost, architectural transparency, and the ability to modify kernel code to serve specialized requirements — impossible with Windows Server — made it the technically and economically optimal choice at scale (Rikap, 2024). By 2017, Linux had achieved 100% penetration among the world's 500 fastest supercomputers, a position it has maintained without interruption — a marker of technical superiority in the most demanding computational environments that is not subject to the marketing ambiguities of broad market share statistics (Pfandzelter & Bermbach, 2024).

The structural lessons of the Linux-Windows server competition are analytically rich. First, the community governance model of Linux — which enabled thousands of contributors across competing corporations to collectively address architectural bottlenecks, security vulnerabilities, and performance limitations far faster than any single vendor could — proved decisively superior in environments where the pace of requirement change exceeded the responsiveness of hierarchical proprietary development (Raymond, 1999; Fitzgerald, 2006). Second, the absence of licensing lock-in allowed cloud providers to build on Linux without incurring the per-server licensing costs that would have made large-scale deployment of Windows Server economically unviable at cloud infrastructure margins — a structural economic advantage that no degree of technical product investment could overcome (West & Gallagher, 2006; Wen & Choudhury, 2019). Third, the modular architecture of Linux — which allows any component to be replaced, patched, or extended without vendor permission — enabled the degree of customization that hyperscale infrastructure requires, from Google's custom kernel patches to AWS's Nitro hypervisor, in ways that a monolithic proprietary operating system structurally cannot accommodate (Pfandzelter & Bermbach, 2024). The Linux-Windows server story is therefore not merely a market share narrative — it is a structural demonstration that community-governed, architecturally modular open source software can outcompete the most well-resourced proprietary product in history when the competitive environment rewards the specific advantages that open source governance and architecture provide.

9.2 Android: Open Source as Proprietary Trojan Horse

Android represents the most commercially significant and most theoretically instructive case study in the open source debate — one whose complexity resists both the celebratory narrative of open source triumph and the critical narrative of proprietary enclosure, because it is simultaneously both (Mayrhofer et al., 2021). As analyzed in Section 5.1, Android's architecture is a layered system in which an open

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

source foundation — the Android Open Source Project, licensed under the Apache License 2.0 — provides the hardware abstraction, kernel, and base libraries, while a proprietary superstructure — Google Mobile Services — provides the application ecosystem, developer APIs, and cloud integration that constitute the commercially meaningful experience for both end users and application developers (Mayrhofer et al., 2021).

The theoretical framework most illuminating for this case is the strategic commoditization logic identified by West and Gallagher (2006): Google released AOSP as open source precisely in order to commoditize the mobile operating system layer, preventing any competitor from achieving a proprietary mobile platform monopoly equivalent to Microsoft's Windows monopoly on the desktop, while retaining the proprietary GMS layer as the locus of competitive differentiation and revenue generation. The strategy was executed with remarkable precision: by making AOSP available under the permissive Apache License — rather than the GPL, which would have required derivative distributions to remain open — Google enabled OEM manufacturers to adopt Android without the obligation to contribute modifications back, while simultaneously ensuring that the commercial viability of those derivatives depended on access to GMS, which only Google controls and which is available only under commercial licensing terms (Mayrhofer et al., 2021; West & Gallagher, 2006).

The geopolitical dimensions of Android's architecture have become salient in the context of trade disputes. When the United States imposed technology export restrictions on Huawei in 2019, the restrictions included denial of access to GMS — which, because Huawei devices outside China depended on GMS for application ecosystem viability, effectively rendered Huawei unable to sell internationally competitive Android devices despite the continuing availability of AOSP (Bechara & Lechner, 2024). This episode demonstrated with unusual clarity the mechanism by which open source at the infrastructure layer provides no protection against proprietary control at the service layer — and confirmed that the question of technological sovereignty in the mobile ecosystem is determined not by who controls the AOSP codebase but by who controls the proprietary service layer that makes the codebase commercially viable (Glasze et al., 2022; Biström et al., 2024). Android is therefore paradigmatic not as a success story of open source democratization but as a demonstration that code openness and power openness are analytically distinct categories — and that the former is compatible with, and can in fact facilitate, the latter's concentration.

9.3 HashiCorp and the License Change: The Day Open Source "Betrayed" Its Community

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

HashiCorp's August 2023 decision to relicense Terraform, Vault, Consul, and Nomad from the Mozilla Public License 2.0 to the Business Source License 1.1 constitutes the most consequential and most thoroughly analyzed open source governance rupture of the current decade — one that exposed structural tensions between commercial sustainability and community governance that had been accumulating throughout the venture-capital-funded open source boom of the 2010s (Bechara & Lechner, 2024; Riehle, 2009).

Terraform had been released as open source under the MPL in 2014 and had, over approximately nine years, accumulated an ecosystem of thousands of users, contributors, vendors, and certified practitioners who had built their technical identities, product offerings, and professional practices on the assumption that Terraform's open source status was a durable commitment rather than a revocable commercial strategy (OpenTofu, 2023). The BSL's restrictions — which prohibit the use of the software to create a product that competes with the licensor's own commercial offerings — targeted specifically the cloud providers and infrastructure vendors who had been offering Terraform-as-a-service or embedding Terraform in commercial platform products, following the pattern of asymmetric value capture identified in Section 3.4 (Bechara & Lechner, 2024; West & Gallagher, 2006). However, the legal ambiguity of the BSL's "competitive use" provisions created uncertainty well beyond those intended targets: organizations whose current or future product roadmaps might conceivably be characterized as competitive with HashiCorp's offerings faced immediate legal exposure, and the inability to predict how HashiCorp might interpret "competitive" in the future made the license change a structural governance risk for any organization that had built critical infrastructure on Terraform (OpenTofu, 2023).

The community response exemplified the fork as a governance mechanism of last resort. Within two weeks of the BSL announcement, a coalition of over 140 companies and 700 individuals formed around the OpenTofu initiative; the Linux Foundation accepted the project as OpenTofu in September 2023; and the first stable production release was available by January 2024 (OpenTofu, 2023; CNCF, 2025). A critical governance accelerant was the CNCF's policy requiring that all tooling used to build CNCF projects — including the build pipelines for Kubernetes and its ecosystem — must be licensed under OSI-approved open source licenses: when HashiCorp's license change rendered Terraform ineligible under this policy, the Linux Foundation had an institutional imperative to support a fork that was entirely independent of any assessment of HashiCorp's business justification (CNCF, 2025). The HashiCorp case teaches three governance lessons of broad applicability. First, community trust in an open source project is a capital asset that can be destroyed discontinuously and may be difficult to rebuild; the relicensing decision damaged HashiCorp's developer

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

community relationships in ways that persisted well beyond the legal resolution. Second, the fork right — the structural protection that open source governance provides against adverse stewardship — is genuinely exercisable when projects are licensed under permissive or copyleft OSI-approved licenses, but is exercisable only once: the fork creates a permanent ecosystem split whose maintenance cost falls on the community rather than the entity that triggered it (O'Mahony, 2007). Third, the pattern of VC-backed open source companies relicensing after achieving commercial scale — described in Section 6.5 — is not a random series of individual decisions but a structural tendency produced by the financial dynamics of venture-backed growth, in which the community-building benefits of open source are most valuable during the growth phase and the revenue-protection benefits of licensing restriction become most attractive after commercial scale is achieved (Cooiman, 2024; Bechara & Lechner, 2024).

9.4 Redis, Elasticsearch, and the War with AWS

The parallel licensing conflicts involving Redis and Elasticsearch in 2018–2024 constitute a connected case study in the economics of open source asymmetric value capture and its governance consequences — and, in the case of both projects, a demonstration that the community response to license restriction has structural consequences that may not serve the original licensor's commercial objectives (Bechara & Lechner, 2024; West & Gallagher, 2006).

The mechanism of the conflict was identical in both cases: MongoDB, Redis, and Elasticsearch had each built their commercial enterprises on open source codebases whose adoption had been driven in significant part by cloud providers — primarily Amazon Web Services — offering managed services based on those codebases at scale, capturing revenue from that adoption without contributing proportionally to the underlying projects' development costs (Bechara & Lechner, 2024). Each company's licensing response — MongoDB's SSPL in 2018, Redis Labs' Commons Clause modifications in 2018, and Elastic's SSPL adoption in January 2021 — was framed by the licensor as a mechanism for ensuring that commercial users who derived economic benefit from the software contribute to its development, but was rejected by the Open Source Initiative as failing to meet the Open Source Definition's criteria of non-discrimination against fields of endeavor (Riehle, 2009). The OSI's statement on Elastic's relicensing characterized the move as harmful to the software commons — noting that external developers had contributed to these projects under the understanding that their work was directed toward the public commons, and that retroactive license changes altered the terms of those contributions without the contributors' consent.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Amazon's response to Elastic's relicensing was strategically significant: rather than accepting the new license terms, AWS announced it would fork the last Apache 2.0-licensed version of Elasticsearch and maintain it independently as OpenSearch, a fully open source project governed through the Apache Software Foundation. The OpenSearch fork attracted contributions from Red Hat, Logz.io, CrateDB, and Aiven, demonstrating that AWS's stated commitment to open source governance was commercially credible enough to anchor a viable community around the fork (Bechara & Lechner, 2024). The Redis case followed an even more complex trajectory: Redis Labs moved from BSD to RSAL/SSPL in March 2024, the Linux Foundation launched the Valkey fork within thirty days of the announcement, and by May 2025 Redis had reversed course by releasing Redis 8 under the open source AGPLv3 — the first major reversal in the relicensing pattern. The balance sheet of the Redis/Elasticsearch licensing conflicts is analytically instructive: the cloud providers retained access to open source versions of both technologies through successful forks; the original companies incurred community trust damage and ecosystem fragmentation without eliminating the managed service competition they sought to curtail; and the net beneficiary was the broader open source ecosystem, which gained well-resourced, foundation-governed alternatives that are structurally more resistant to single-vendor governance capture than the projects they replaced (O'Mahony, 2007; West & Gallagher, 2006).

9.5 Meta and LLaMA: Open Source as Market Strategy

Meta's decision to release the weights of its LLaMA series of large language models beginning in February 2023 — and to progressively expand the permissiveness of its licensing terms through subsequent releases — represents the most consequential open source strategic action in the AI domain to date, and the one that most clearly illustrates the corporate commoditization logic analyzed in Sections 3.3 and 7.2 (Widder et al., 2023; Rikap, 2024).

The question of whether Meta's LLaMA releases are altruistic or strategic admits only one analytically credible answer: they are strategic, and their strategic logic follows directly from the framework developed by West and Gallagher (2006). Meta's core business — advertising revenue derived from engagement on its social platforms — does not depend on and is not enhanced by maintaining proprietary control over AI model weights. What it does depend on is: an AI ecosystem that allows Meta to deploy competitive AI features across its consumer platforms at low marginal cost; a developer and research community that builds on Meta's AI tools and frameworks, strengthening the attractiveness of Meta's broader infrastructure and cloud partnerships; and the commoditization of AI model capabilities, which reduces the competitive advantage of OpenAI, Anthropic, and Google — whose closed, API-only

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

models constitute a business model dependent on scarcity of frontier capability that Meta's open-weight releases systematically undermine (Rikap, 2024; West & Gallagher, 2006). The structural parallel to Google's AOSP release is precise: just as Google released Android to commoditize the mobile operating system layer while retaining control at the services layer, Meta releases LLaMA to commoditize the AI model layer while retaining competitive advantage in its advertising infrastructure, data assets, and consumer platform distribution (Rikap, 2024).

The impact on the broader AI ecosystem has been substantial and genuinely democratizing in one specific and important sense: the availability of open-weight frontier-capable models has made it possible for organizations without the capital to pay OpenAI's or Anthropic's API fees to deploy capable AI systems for sensitive or regulated use cases where routing data through a third-party proprietary API is not permissible — healthcare, defense, finance, and government contexts in which data sovereignty requirements are paramount (European Commission, 2025; Widder et al., 2023). The Llama 3.1 405B release in July 2024 — described by Meta as the first frontier-level open-weight model — reduced API inference costs across the industry by forcing competing providers to lower prices in response to a freely available alternative (European Commission, 2025). However, the governance limitations of Meta's open-weight releases — identified in Section 4.5 and 5.6 — remain structurally significant: the training data, training code, and data pipeline are not disclosed; the Llama license imposes commercial restrictions on large-scale deployments by organizations above a defined user threshold; and Meta retains unilateral authority over licensing terms for future versions (Widder et al., 2023; ACLU, 2025). These limitations mean that LLaMA occupies the same ambiguous position as Android in the open source taxonomy: it is sufficiently open to drive ecosystem adoption and community contribution, while retaining sufficient proprietary control to ensure that the value created by that ecosystem ultimately accrues to Meta's strategic interests.

9.6 Kubernetes: When Open Source Becomes an Industry Standard

Kubernetes represents the clearest contemporary example of an open source project achieving categorical dominance — not merely market leadership, but the definition of a technical category itself — and the CNCF governance model that enables that dominance provides the most instructive template available for understanding how multi-stakeholder open source governance can operate at industrial scale (CNCF, 2025).

Kubernetes originated as an internal Google container orchestration system derived from Google's proprietary Borg infrastructure management platform. Google's 2014 decision to release it as open source and donate it to the newly formed CNCF in 2015

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

followed the strategic commoditization logic described throughout this analysis: by open-sourcing a container orchestration technology that it had developed internally, Google commoditized a capability that competitors might otherwise have developed as proprietary differentiators — while positioning itself as the authoritative contributor and cloud provider for Kubernetes-based workloads (Rikap, 2024; West & Gallagher, 2006). The transfer of operational control to the community, formally completed in August 2018, was essential to the project's subsequent adoption trajectory: organizations that might have been reluctant to build critical infrastructure on a Google-controlled project were able to adopt it as a genuinely community-governed standard whose roadmap no single vendor could unilaterally control (CNCF, 2025; Cosentino et al., 2020).

The CNCF governance model has become itself a paradigm for neutral multi-stakeholder open source governance. Its project lifecycle — sandbox, incubating, and graduated maturity levels — provides a structured framework for evaluating project health, community governance, and production readiness that enables organizations to make adoption decisions with confidence about long-term sustainability (CNCF, 2025). Its requirement that any tooling used to build CNCF projects must be licensed under OSI-approved licenses creates an institutional norm of openness that has had direct governance consequences — most visibly in the OpenTofu fork, where the CNCF's toolchain openness policy created institutional imperative for a Terraform alternative when HashiCorp's relicensing made Terraform ineligible (CNCF, 2025). By 2024, Kubernetes had achieved 80% production adoption, growing at 20.7% year over year; 93% of surveyed organizations were using, piloting, or actively evaluating it; and its contributor base had grown to over 88,000 contributors from more than 8,000 companies across 44 countries — a scale of multi-stakeholder participation that has no precedent in the history of software development (CNCF, 2025). The categorical lesson of Kubernetes is that open source governance, when structured through genuinely neutral multi-stakeholder institutions, can produce infrastructure standards with the adoption breadth and longevity previously achievable only by standards bodies or de facto monopolists — and can do so at a pace of innovation that neither can match.

10. The Debate in Specific Sectors

10.1 Healthcare

10.1.1 Electronic Health Records: Open Source vs. Proprietary

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

The healthcare sector presents one of the most consequential applications of the open source versus proprietary software debate, because the software systems at stake — electronic health record (EHR) systems — manage data that is simultaneously among the most sensitive categories of personal information and among the most critical inputs to clinical decision-making. The structural stakes of the open/proprietary choice in healthcare therefore extend beyond the organizational to encompass patient safety, equity of access to care, and the long-term integrity of population health data (Shaikh et al., 2022).

The market for EHR systems in high-income countries is dominated by proprietary vendors — principally Epic and Oracle Cerner in the United States and comparable enterprise systems in Europe — whose products offer extensive regulatory certification, vendor support, and integration with clinical workflows at costs that place them beyond the reach of most health systems in low- and middle-income countries (Shaikh et al., 2022). A systematic review of the most recent advances in open source EHR systems identifies OpenMRS, OpenEMR, GNU Health, and Bahmni as the most functionally rich open source alternatives, finding that these systems have reached sufficient architectural maturity to provide integrated clinical data management, role-based security, HL7 FHIR interoperability, and standards-compliant documentation functionality (Shaikh et al., 2022). A broader systematic review of the utilization of open source EHR systems across all continents found that open source EHRs have been widely adopted in resource-limited regions, particularly in sub-Saharan Africa and South America, where they provide opportunities to improve national healthcare levels with minimal financial resources as a solution to the high cost and inflexibility of proprietary health information systems (Jawhari et al., 2016).

OpenMRS — the Open Medical Record System — is the paradigmatic case of open source infrastructure achieving genuine population-scale healthcare impact in low-resource settings. Originally developed in 2004 by researchers at the Regenstrief Institute and Partners In Health in response to the HIV epidemic in Kenya, OpenMRS has grown into a multi-institutional, non-profit collaborative that currently operates across approximately 6,500 healthcare facilities in over 40 countries and manages the records of approximately 14.6 million patients (Jawhari et al., 2016). The system's modular architecture, which allows new data items, forms, and reports to be added without modifying the core codebase, and its adherence to HL7 and OpenEHR standards have enabled the large-scale adaptation required for deployment across diverse clinical contexts and disease programs (Shaikh et al., 2022).

10.1.2 Vendor Lock-In and Health Data

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

The risk of vendor lock-in in health data systems is structurally more severe than in most enterprise software domains because patient data is not merely operationally critical but clinically indispensable: a health system that cannot migrate its patient records from a proprietary EHR is genuinely captive to that vendor's pricing, support, and product continuity decisions in a way that has direct implications for patient care (Petcu et al., 2013). The proprietary EHR market's concentration — Epic alone serves over 36% of U.S. hospital patients — creates structural conditions in which vendor pricing power, proprietary data formats, and interoperability restrictions impose significant costs on health systems that seek to change vendors or to build population health analytics capabilities across multiple systems (Shaikh et al., 2022). Open source EHR systems, and the HL7 FHIR interoperability standard that both open source and progressive proprietary systems increasingly implement, represent structural responses to this lock-in risk: by standardizing the data representation layer, FHIR creates a degree of portability that reduces the proprietary data format dependency even where the application layer remains closed (Jawhari et al., 2016; Shaikh et al., 2022).

10.1.3 Regulation and Certification as Barriers

A structural asymmetry between open source and proprietary EHR adoption in high-income countries is created by regulatory certification requirements. In the United States, EHR systems must obtain certification under the Office of the National Coordinator for Health Information Technology criteria to qualify for federal incentive programs and to satisfy meaningful use requirements for healthcare providers receiving Medicare and Medicaid reimbursement. OpenEMR is the only widely deployed open source EHR to have obtained ONC certification, meaning that most open source alternatives are structurally ineligible for the regulatory incentive structure that drives EHR adoption in the United States, regardless of their technical capabilities (Shaikh et al., 2022). This regulatory asymmetry does not reflect a technical inadequacy of open source EHR systems, but rather the resource requirements of the certification process itself — which demands sustained investment in documentation, testing infrastructure, and regulatory expertise that commercial vendors with dedicated compliance teams are better positioned to sustain than volunteer-governed open source projects (Jawhari et al., 2016).

10.2 Education

10.2.1 Moodle vs. Blackboard: The LMS Battle

The learning management system market provides a longitudinal empirical case study in the open source versus proprietary competition that has unfolded across

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

more than two decades and produced a complex, non-linear outcome that defies simple characterization. Blackboard, founded in 1997 as one of the first commercial LMS platforms, dominated the early market through an aggressive acquisition strategy that successively absorbed WebCT, ANGEL Learning, Elluminate, and eventually Moodlerooms — a partnership that the open source community received with considerable alarm, given concern about what Blackboard might do with Moodle products. Moodle, developed by Martin Dougiamas and first released in 2002 under the GNU General Public License, was designed explicitly around constructivist pedagogical principles and has grown into the most widely adopted open source LMS globally (Zanjani, 2022). A systematic review of trends in Moodle usage, analyzing 155 peer-reviewed journal articles from 55 countries published between 2015 and 2021, found that Moodle is the most popular and preferred open source LMS, with a high rate of acceptance across institutions of all sizes and types, used in 53% of reviewed articles for curriculum development including learning modules and assessments for blended and online courses (Zanjani, 2022).

A comparative study of the two leading LMS platforms found that both offer functionally comprehensive sets of communication, productivity, and student involvement tools, but that Moodle's open source architecture provides significant advantages in customizability, localization, and total cost of ownership, while Blackboard's proprietary model provides advantages in integrated vendor support and enterprise compliance tooling (Momani, 2010). The broader competitive dynamics of the LMS market have shifted significantly since these early comparisons: Canvas, owned by Instructure, has captured substantial market share in North American higher education; Blackboard's global market share declined from approximately 21% to approximately 12% between 2012 and 2021 (Lemenager, 2022). The LMS competition illustrates the general pattern of open source adoption analyzed throughout this article: in cost-sensitive contexts — including public education systems in developing countries and smaller higher education institutions — the open source model's zero licensing cost and community-supported customization capacity provides decisive advantages; in large research universities and enterprise corporate learning contexts, proprietary platforms' integrated support and compliance tooling maintains competitive relevance (Ven & Verelst, 2008).

10.2.2 Open Educational Resources and Sustainability

Open Educational Resources (OER) — the extension of open source principles to educational content — represent the natural pedagogical complement to open source LMS platforms. The UNESCO Recommendation on OER, adopted unanimously in November 2019, committed member states to supporting the creation, adaptation, and redistribution of freely available, openly licensed educational content across all

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

levels of education, explicitly framing OER as a mechanism for addressing global educational equity (UNESCO, 2019). Research on OER adoption in higher education identifies barriers at individual, institutional, and policy levels that closely mirror the barriers to open source software adoption identified in Section 2.3: the invisible complexity barrier manifests as the additional effort required to find, evaluate, adapt, and integrate OER into course design; the sustainability barrier manifests as the absence of a business model that ensures OER repositories remain funded and updated over time (Olivier, 2021).

10.3 Government and the Public Sector

10.3.1 Open Source First Policies

The adoption of open source software policies in national and subnational governments represents one of the most significant institutional developments in the open source landscape of the past decade. The United Kingdom's Government Service Standard, which has included an "open source first" preference since 2014, combined with the Government Digital Service's systematic development of open source code published on GitHub, has established the UK as one of the most advanced practitioners of open source government. France's Socle Interministériel de Logiciels Libres provides an annually updated catalogue of recommended open source software for public administration use, institutionalizing open source preference within the procurement framework of the French state. Germany's Online Access Act 2.0, agreed in February 2024, is among the most explicit — it mandates that government entities prefer open source software over licensed alternatives that restrict use, distribution, transfer, or modification (Neudert, 2024).

The academic argument for open source in government extends beyond cost to encompass a transparency principle with democratic implications: code that implements or mediates public decisions and services should be publicly auditable, because citizens and legislators have a legitimate interest in understanding how algorithmic systems embedded in government processes function (Tai, 2021; Hong & Ji, 2024). This argument has become particularly salient as governments deploy algorithmic decision-making systems in areas including benefits eligibility, risk scoring, and public procurement — systems that, if built on proprietary software, cannot be independently audited for discriminatory bias or systematic error (Hong & Ji, 2024). The principle that public money should produce public code — articulated by the Free Software Foundation Europe and endorsed by the European Commission — is grounded precisely in this democratic accountability argument: software developed with public funds should be available for public use, modification, and scrutiny (Neudert, 2024).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Research measuring the scale of U.S. federal government investment in open source software finds that the 2021 federal OSS investment was estimated at approximately $407 million across seventeen major agencies, with the Department of Energy, NASA, and the General Services Administration as the most active contributors to publicly available government codebases (Shrivastava & Korkmaz, 2024). The U.S. Federal Source Code Policy, established in 2016 and codified through Code.gov, represents an institutional commitment to open source production and sharing within the federal government that is systematically documented and measurable in ways that earlier open government rhetoric was not (Shrivastava & Korkmaz, 2024).

10.3.2 Brazil's E-Government Trajectory

Brazil's e-government trajectory illustrates the ambivalence of national open source policy in practice. Beginning with the e-government program launched in the early 2000s, Brazil developed a formal policy preference for open source software in public administration — embodied in the work of SERPRO (the Federal Data Processing Service), the e-gov portal, and successive iterations of the national software freedom policy. The Brazilian open government data trajectory, beginning with the e-government program approximately twenty-five years ago, generated successive layers of open data policy and infrastructure, most recently manifested in the AI Plan 2024–2028 with its emphasis on indigenous open source AI development (Santos et al., 2024). However, the policy record includes significant reversals: institutional priorities have oscillated between open source advocacy and pragmatic procurement of proprietary platforms, reflecting the political instability of technology policy in the absence of stable legislative foundations (Santos et al., 2024).

10.4 Finance and Fintechs

10.4.1 Open Banking as Regulatory-Forced Openness

Open banking — the regulatory framework that requires incumbent banks to share customer data with authorized third-party providers through standardized APIs — represents a distinctive form of openness that is imposed by regulatory mandate rather than market choice, and that creates conditions for competition that the incumbent institutions would not voluntarily have created (Xie & Hu, 2024). The EU's Payment Services Directive 2 (PSD2), which entered into force in 2018, and analogous frameworks in the United Kingdom, Australia, Brazil (Open Finance regulation through the Banco Central do Brasil), and India (Account Aggregator framework) have collectively created a global regulatory architecture of data portability in financial services that parallels, in regulatory logic, the interoperability requirements of the EU Data Act for cloud services (Eloul et al., 2024). Research

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

reviewing the emerging open banking literature finds that empirical studies from the UK demonstrate that open banking facilitates consumer access to a broader array of financial services, that non-bank fintech entities gain competitive advantages through data sharing, and that the overall welfare effects are positive through market entry and competitive pressure effects (Xie & Hu, 2024). The open banking framework does not mandate open source software per se, but its API standardization requirements create an interoperability infrastructure that significantly reduces the vendor lock-in effects of proprietary banking system architectures (Eloul et al., 2024).

10.4.2 Open Source Core Banking

In the domain of core banking systems — the transaction processing infrastructure that constitutes the operational heart of financial institutions — open source alternatives remain largely confined to developing country and microfinance contexts, with mainstream Western retail banking dominated by a small number of large proprietary vendors. Apache Fineract, maintained by the Apache Software Foundation and originally developed by the Mifos Initiative as a core banking platform for microfinance institutions, has achieved deployment across approximately 400 institutions in 37 countries and serves more than 20 million customers (Apache Software Foundation, 2024). Its architecture — modular, API-first, built on Java and Spring Boot with PostgreSQL or MySQL persistence — provides the technical foundation for digital financial services across diverse deployment contexts, from mobile money platforms in sub-Saharan Africa to cooperative banks in Southeast Asia (Apache Software Foundation, 2024). However, regulatory requirements in high-income financial markets constitute a structural barrier to open source core banking adoption that mirrors the EHR certification barrier in healthcare: regulators in the United States and Europe require validated, audited software builds with defined liability structures and formal change management processes that the distributed, volunteer-governed development model of projects like Apache Fineract cannot easily satisfy (Xie & Hu, 2024).

10.4.3 Security as a Proprietary Argument

The financial sector's security argument for proprietary software is among the strongest domain-specific cases for closed development in this article. Financial institutions are subject to regulatory frameworks — including PCI-DSS for payment card security, SOC 2 for cloud service providers, and the EU's Digital Operational Resilience Act (DORA), which entered into force in January 2025 — that impose specific requirements around software change management, incident response, and supply chain risk management that are more tractable for proprietary vendors with contractual accountability than for community-governed open source projects

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

(Anderson, 2020). The opacity argument, while generally disfavored in security research following Kerckhoffs's principle (Section 5.7), has genuine relevance in financial fraud prevention: anti-fraud logic, transaction monitoring rules, and risk scoring models that are publicly visible can be reverse-engineered by adversaries seeking to circumvent them (West & Gallagher, 2006).

10.5 Artificial Intelligence and Big Tech

10.5.1 The Corporate Open Source Cycle

The behavior of large technology corporations with respect to open source AI follows a structural pattern that generalizes the strategic commoditization logic analyzed throughout this article and achieves its clearest expression in the AI domain. The cycle proceeds in four stages: first, a corporation invests in developing a new AI capability for internal use, building proprietary competitive advantage at the frontier; second, the capability matures and begins to be replicated by competitors, reducing its proprietary value; third, the corporation releases the capability as open source, commoditizing it, attracting community contributions that reduce its own maintenance burden, and simultaneously building developer ecosystem loyalty; fourth, it retains proprietary control at the frontier — the next-generation capability, the fine-tuning infrastructure, or the managed cloud service — where competitive differentiation now resides (Rikap, 2024; West & Gallagher, 2006). Google's 2015 release of TensorFlow exemplified this cycle precisely: TensorFlow had been Google's internal machine learning framework since 2011; its open-sourcing in November 2015 came at the point when Google had already moved its internal research to a more advanced successor, and it commoditized the deep learning framework layer at the expense of any competitor who might have built a proprietary equivalent (Rikap, 2024). PyTorch, developed by Meta AI Research and open-sourced in 2017, was similarly released at a point of internal maturity, attracting the research community's adoption through its dynamic computation graph architecture — a technical choice that proved better adapted to the experimental iteration patterns of academic ML research — and subsequently eclipsing TensorFlow in research dominance, forcing Google to adopt a more PyTorch-compatible API in TensorFlow 2.0 (European Commission, 2025).

10.5.2 The Framework War That Open Source Won Twice

The PyTorch/TensorFlow competition illustrates an unusual double-open source dynamic. Both frameworks are open source; the competition between them is therefore not a classic open source versus proprietary confrontation but a competition between two different open source governance and technical design philosophies. TensorFlow's governance was tightly controlled by Google, with major architectural

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

decisions made internally and released to the community as fait accompli — a pattern consistent with the strategic commoditization model but misaligned with the community governance expectations of academic researchers who wanted influence over the framework's direction (Rikap, 2024). PyTorch's governance was more responsive to community feedback, a characteristic that — combined with its superior usability for research workflows — drove its adoption to a point at which it became the de facto standard for AI research, subsequently followed by industry practitioners trained on it (European Commission, 2025). The lesson of the framework war is that within the open source ecosystem, governance quality and community responsiveness function as competitive advantages that can determine outcomes even when the competing artifacts are both open source — extending the governance analysis of Section 7.1 into the domain of AI infrastructure.

10.5.3 Hugging Face: New Gatekeeper Risk

Hugging Face has emerged as the central distribution platform for the open source AI ecosystem — the functional equivalent of GitHub for model weights, datasets, and AI code. Its Model Hub hosts hundreds of thousands of models; its Datasets Hub provides the training data for thousands of research projects; its Transformers library has become the standard programmatic interface for working with language models across frameworks; and its Spaces platform enables researchers to deploy and share interactive model demonstrations (European Commission, 2025). This centrality is a dual-edged achievement: Hugging Face has genuinely democratized access to AI infrastructure, providing small organizations, academic researchers, and individual developers with access to state-of-the-art model capabilities that would otherwise require substantial cloud API budgets (European Commission, 2025). However, the concentration of the open source AI ecosystem in a single private company introduces a structural centralization risk that mirrors the concentration risks identified throughout this analysis in other domains. Research on the political economy of open AI identifies the risk that large platforms could capture the open source AI ecosystem by making their commercial services indispensable to AI development in ways that gradually erode the independence that open source is supposed to provide (Widder et al., 2023). Hugging Face is a venture-backed private company with financial return obligations; if its commercial trajectory leads to the same enshittification dynamic identified in Section 6.5 — progressive restriction of features, pricing of previously free services, or acquisition by a large technology corporation — the open source AI ecosystem's most critical distribution infrastructure would face the same governance risk that the hashicorp, Redis, and Elasticsearch episodes illustrated for open source software (Widder et al., 2023; Doctorow, 2023).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

11. Trends and the Future of the Debate

11.1 The Inevitable Convergence: Hybrid Models Dominate

The analytical trajectory of this article converges on a conclusion that the empirical evidence across every sector and domain examined makes difficult to resist: the binary opposition between open source and proprietary software, while analytically necessary as a conceptual anchor, does not accurately describe the organizational and commercial reality of most software ecosystems in the mid-2020s. The dominant organizational forms are hybrid — open core architectures, SaaS services built on open source foundations, dual licensing arrangements, and corporate contribution programs that blend open and closed elements within single firms — and this hybridization is accelerating rather than receding (Weiss et al., 2022; West & Gallagher, 2006).

The structural drivers of convergence are traceable across the institutional, economic, and technological dimensions analyzed throughout this article. At the institutional level, the corporate open source paradox — in which firms contribute to open source projects they also depend on commercially — has become so pervasive that it is now the rule rather than the exception: virtually every major technology corporation simultaneously maintains proprietary products, contributes to open source foundations, and builds commercial services on top of open source infrastructure (Rikap, 2024; Dahlander & Magnusson, 2008). At the economic level, the commercial success of open core and SaaS-on-open-source models has demonstrated that sustainable revenue generation is possible without requiring either fully proprietary or fully open codebases — eliminating the economic pressure that once forced firms toward the poles of the spectrum (Weiss et al., 2022; Riehle, 2009). At the technological level, the emergence of AI-assisted development, cloud-native infrastructure, and managed services has shifted the competitive frontier from the software layer — where open source has largely won — to the service, data, and intelligence layers, where proprietary differentiation persists regardless of the openness of the underlying code (Widder et al., 2023; Bechara & Lechner, 2024).

What emerges in the place of the simple open/closed dichotomy is a more nuanced landscape of openness gradients: the fundamental question is no longer whether a given technology is open or proprietary, but rather at which layer of the stack openness is provided, under what governance conditions, with what resource commitments, and in whose interests — a set of questions that the theoretical frameworks developed in this article are designed to address (Pohle, 2024; O'Mahony, 2007).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

11.2 Regulation as a Determining Factor

11.2.1 The EU AI Act: Does It Favor Open Source?

The EU AI Act (Regulation 2024/1689), which entered into force on August 1, 2024, represents the most consequential regulatory intervention in the governance of AI systems in history — and its interaction with the open source versus proprietary debate is nuanced in ways that defy simple summary. The Act establishes a risk-based framework in which AI systems are classified as unacceptable risk (prohibited), high risk (subject to stringent obligations), limited risk (subject to transparency requirements), and minimal risk (unregulated). For general-purpose AI models, the Act imposes transparency obligations on all providers, including requirements to publish a sufficiently detailed summary of training data, adopt a copyright compliance policy, and provide technical documentation to downstream providers. For GPAI models presenting systemic risks — defined as those requiring more than 10^{25} FLOPs to train — additional safety evaluation and incident reporting obligations apply (European Commission, 2024; Regulation EU 2024/1689).

The Act's treatment of open source AI is materially favorable: GPAI models released under free and open-source licenses are exempt from most GPAI obligations unless they present systemic risks. This exemption reflects a legislative judgment that the transparency already provided by open-weight release constitutes a sufficient contribution to the Act's accountability objectives — though the Open Future observatory notes that the exemption conditions, clarified through a European Commission consultation document published as part of the GPAI guidelines preparation in early 2026, require careful compliance analysis to ensure that open source labels genuinely meet the required disclosure standards rather than functioning as an evasion of regulatory transparency requirements (Open Future, 2025). The broader transparency orientation of the AI Act — its requirements for training data documentation, technical disclosure, and human oversight — structurally favors software architectures that are amenable to independent audit, creating a regulatory environment in which the auditability advantages of open source become compliance assets rather than merely engineering preferences (Mügge, 2024; Biström et al., 2024).

11.2.2 The Cyber Resilience Act: A Self-Defeating Regulation?

The EU Cyber Resilience Act (Regulation 2024/2847), which entered into force on December 10, 2024, with full application beginning December 11, 2027, imposes mandatory cybersecurity requirements on all products with digital elements placed on the European market. Its genesis in the landscape of unpatched consumer IoT

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

devices and vulnerable connected infrastructure described in Section 5.1 is clear and its objective legitimate; its interaction with the open source ecosystem, however, was initially potentially catastrophic and remains a concern even after significant community-led amendments (Aliprandi & Muselli, 2024). The initial draft of the CRA would have imposed on any developer or maintainer of open source software that could be incorporated into commercial products the same compliance obligations — incident reporting, vulnerability disclosure timelines, risk assessments — as the commercial manufacturers who profit from that software, creating liability exposure for volunteer maintainers with no commercial relationship to the products built on their code and no resources to satisfy bureaucratic compliance requirements (Aliprandi & Muselli, 2024). The open source community's response was unified and forceful: the Eclipse Foundation, Open Source Initiative, Linux Foundation, and over 100 organizations signed an open letter warning of a chilling effect on open source development globally if the Act were implemented as written.

The legislative response preserved most of the open source ecosystem's core functioning: the final text introduces a category of open source steward — foundations, non-profit associations, and similar entities — whose role is acknowledged in the regulatory framework and for which a lighter compliance regime applies; non-commercial open source software is exempted from the Act's requirements; and projects receiving donations that do not seek profit from those donations are exempt from what constitutes supply in the course of commercial activities (Aliprandi & Muselli, 2024). However, the distinction between commercial and non-commercial activity, and between a project maintainer and a commercial steward, remains legally untested and potentially contentious in application — and the requirements imposed on commercial integrators of open source components, including mandatory upstream vulnerability reporting, may have second-order effects on open source communities' ability to manage disclosure timelines on their own terms (Aliprandi & Muselli, 2024).

11.2.3 GDPR, LGPD, and the Auditability Argument

The General Data Protection Regulation (GDPR, EU 2016/679) and Brazil's Lei Geral de Proteção de Dados (LGPD, Law 13.709/2018) share a structural logic that creates alignment with the open source transparency principle: both frameworks impose obligations of accountability, transparency, and demonstrability that require organizations to be able to explain, audit, and document how personal data flows through their systems. This requirement is structurally easier to satisfy with open source software, whose code can be independently audited to verify that it implements data handling in the manner claimed — a capability that proprietary black-box software systems cannot as readily provide (Pohle, 2024; Santos et al., 2024).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Data protection authorities in several jurisdictions have issued guidance indicating that the use of open source components in data processing systems facilitates compliance demonstration — though neither GDPR nor LGPD explicitly mandates open source, and the auditability advantage of open source is conditional on the existence of actual, resourced audit capacity, as the Heartbleed episode demonstrated (Walden, 2020).

11.3 AI and the Redefinition of "Open"

The single most consequential intellectual challenge posed by the AI era to the open source movement is the fundamental question of what "open" means when the product is not a piece of executable code but a statistical model trained on data. This question is not merely definitional: it goes to the core of what properties — transparency, reproducibility, auditability, community governance — the concept of openness is intended to secure, and whether those properties can be realized through weight disclosure alone or require something more (Widder et al., 2023).

The Open Source Initiative's Open Source AI Definition version 1.0, published in October 2024 after two years of development in collaboration with researchers, lawyers, policymakers, and industry representatives, establishes four essential freedoms for Open Source AI — to use, study, modify, and share the system for any purpose — and specifies that realizing these freedoms requires access to three components: (1) complete source code used to train and run the system, under OSI-approved licenses; (2) model parameters including weights and configuration settings, under OSI-approved terms; and (3) data information — sufficiently detailed information about the data used to train the system so that a skilled person can build a substantially equivalent system (OSI, 2024). Crucially, and controversially, the OSAID does not require the training data itself to be made available, accepting instead a disclosure standard for data description that stops short of data access. Critics from the AI research community have characterized this as leaving a gap that leaves the definition practically weakened: if the training data is not accessible, the ability to reproduce, verify, or genuinely understand the model's behavior is severely constrained regardless of weight and code availability (OSI, 2024). The practical consequence of the OSAID's standard is that the list of AI systems that fully satisfy it is short: Pythia (EleutherAI), OLMo (AI2), and the LLM360 models (Amber, CrystalCoder) are among the few that qualify — none of the major commercially deployed open-weight models, including Llama 3, Mistral, and Falcon, meet the OSAID criteria (OSI, 2024; Widder et al., 2023). The OSAID itself has been received critically from multiple directions: Liesenfeld & Alikhani (2024) argue that it sets the disclosure bar too low to enable genuine scientific reproducibility, while Contractor et al. (2023) caution that mandating full data disclosure may conflict with privacy law

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

and produce its own harms. The international dimension adds further complexity: governance frameworks adequate for Global North research contexts may be poorly adapted to the resource constraints and regulatory environments of Global South AI developers (Jiang & Belli, 2024).

The question of whether truly open AI models — satisfying the OSAID's full criteria including meaningful data information disclosure — are viable at scale is an open empirical and economic question. The principal structural obstacle is the competitive sensitivity of training data: as noted in Section 9.5, the methods of assembling, filtering, and labeling training data constitute a core competitive advantage for AI developers, and their disclosure would reduce the barrier to entry for competitors in ways that the disclosure of model weights alone does not. A secondary obstacle is the legal complexity of training data provenance: much of the data on which frontier AI models are trained consists of copyrighted material whose status under fair use or equivalent doctrines is contested in ongoing litigation, and full disclosure of training data details could increase the legal exposure of model developers in ways that make compliance with the OSAID financially hazardous (OSI, 2024; Widder et al., 2023).

11.4 Decentralization and Web3

The blockchain and Web3 movement represents the most ambitious attempt yet to extend open source principles beyond software code to the entire architecture of digital platforms — creating systems in which governance rules are encoded in publicly auditable smart contracts, economic incentives are mediated through transparent token mechanisms, and the power asymmetries of centralized proprietary platforms are structurally eliminated through distributed infrastructure (Rozas et al., 2021). The theoretical appeal of this vision is genuine and well-grounded in the Ostromian governance framework analyzed in Section 3.4: blockchain-enabled Decentralized Autonomous Organizations present a set of affordances for the governance of digital commons — tokenization, self-enforcement of rules through smart contracts, transparent collective decision-making — that could, in principle, address structural weaknesses of traditional open source governance (Rozas et al., 2021).

The empirical reality of blockchain governance has, however, consistently fallen short of these theoretical promises. Governance token concentration — in which a small number of large token holders exercise disproportionate control over DAO governance decisions — has been documented as a pervasive structural feature of decentralized governance systems; empirical analysis across 30,000 DAOs found that 53% were inactive with no proposals in six months, and that voter turnout decreased as DAO size increased, with average voter participation per proposal in

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

some systems below 1% (Cong et al., 2025). DAOs face a trilemma between autonomy, decentralization, and efficiency — with empirical evidence revealing persistent centralization in practice due to the concentration of voting power among large token holders and the off-chain coordination advantages enjoyed by venture capital networks and large investor groups (Cong et al., 2025). The risk of corporate capture of nominally decentralized blockchain projects is not hypothetical: the history of the cryptocurrency sector includes numerous cases in which founding teams, large token holders, or venture capital backers have exercised effective control over governance processes that were structurally described as decentralized, producing outcomes more consistent with centralized decision-making than community governance (Cong et al., 2025; Jansen et al., 2021). The complexity and technical inaccessibility of blockchain governance mechanisms constitutes a further democratization barrier: meaningful participation in on-chain governance requires technical literacy, economic resources to hold governance tokens, and availability to monitor and respond to governance proposals — a set of requirements that structurally excludes the populations that the democratizing rhetoric of Web3 claims to serve.

11.5 The Emerging Role of the Global South

The structural position of countries in the Global South, characterized by high consumption and low code contribution (Jiang & Belli, 2024; Jiang et al., 2024). Countries in the BRICS grouping and across the broader developing world are overwhelmingly users of open source infrastructure developed and governed primarily by organizations in the United States, Europe, and a small number of East Asian economies — participating in the benefits of open source without participating in the governance decisions that shape what gets built, how it is governed, and whose interests it serves (Jiang & Belli, 2024; Jiang et al., 2024).

The structural causes of this contribution gap are well documented: language barriers that make English-dominant community governance processes inaccessible to developers whose primary language is Portuguese, Hindi, Indonesian, or other non-English languages; infrastructure gaps in high-speed internet connectivity and development tooling that increase the friction cost of open source participation; economic conditions that preclude the sustained investment of volunteer time in unpaid open source contribution that the traditional model assumes; and a historical concentration of open source governance in a small number of large technology companies and foundations headquartered in the Global North whose priorities reflect the needs of their primary stakeholders (Jiang & Belli, 2024; van Dijk, 2020).

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Three recent developments suggest that this asymmetry may be beginning to shift, though the magnitude and durability of the shift remain uncertain. First, digital sovereignty initiatives across the Global South — including India's India Stack, Brazil's Digital Public Infrastructure program, and analogous programs in Indonesia and South Africa — have created institutional contexts in which open source software development receives state-level funding and strategic priority (Jiang & Belli, 2024; Srivastava & Mishra, 2025). The India Stack's UPI payment system and Aadhaar digital identity infrastructure represent open source-based digital public infrastructure developed by Indian institutions, deployed at a scale — 1.5 billion potential users — that dwarfs any comparable deployment in the Global North, and available for adoption by other developing countries seeking similar capabilities (Jiang & Belli, 2024). Brazil's Pix payment system, developed by the Banco Central do Brasil and deployed in 2020, similarly is consistent with the interpretation that public institutions in the Global South can develop and deploy digital infrastructure at population scale, with Brazilian leadership of the G20 Digital Economy Working Group in 2024 advancing a Digital Public Infrastructure agenda specifically designed to transfer these capabilities to lower-income countries (Belli, 2024). Second, the AI sovereignty moment has created new incentive structures for open source AI development in the Global South: the prohibitive cost of proprietary frontier AI API access, combined with the strategic concern about dependence on U.S. AI platforms articulated in national AI strategies including Brazil's AI Plan 2024–2028, has created both the motivation and, increasingly, the institutional support for indigenous open source AI model development (Institute for Global Change, 2026; European Commission, 2025). Third, the growing importance of multilingual and multimodal AI capabilities has created a domain in which Global South developers have a comparative advantage — Portuguese, Hindi, Swahili, and Indonesian language AI capabilities are poorly served by English-centric proprietary models, and the open source AI ecosystem provides a path for domestic research communities to address this gap through fine-tuning and community-governed development (European Commission, 2025; Jiang & Belli, 2024).

The strategic opportunity for Global South countries in the open source landscape extends beyond consumption and sovereignty to participation in standard-setting. The OSI's development of the Open Source AI Definition, the CNCF's cloud native governance frameworks, and the governance structures of major open source foundations represent processes through which the norms of the digital commons are established — and in which Global South voices have historically been underrepresented. As digital public infrastructure programs in Brazil, India, and analogous countries generate indigenous open source development at scale, the institutional conditions for meaningful Global South participation in open source

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

governance are being created, albeit gradually (Belli, 2024; Santos et al., 2024). The opportunity is not merely to contribute to the standards that others define, but to define standards that serve the needs of the majority of the world's population — a project whose realization would represent the deepest possible fulfillment of the open source movement's founding democratic aspirations (Jiang & Belli, 2024; Pohle, 2024).

12. Conclusion

The analytical trajectory of this article arrives at a conclusion that the accumulated evidence across eleven sections makes difficult to contest: there is no absolute winner in the conflict between open source and proprietary software. The question "which model is better?" is not merely unanswerable in the abstract — it is the wrong question. The productive reformulation, which this article has sought to operationalize throughout, is contextual and distributional: better for whom, under what institutional conditions, for which use case, at which layer of the technology stack, and with what governance arrangements? The answers to these questions vary systematically across the domains, sectors, and historical periods examined, and they point toward a landscape of structured trade-offs rather than a hierarchy of models (West & Gallagher, 2006; Weiss et al., 2022; Linåker et al., 2025).

The evidence assembled across the technical, economic, organizational, and political dimensions analyzed in this article supports the following synthesis. Open source software has achieved categorical dominance at the infrastructure layer of the digital economy: server operating systems, container orchestration, AI development frameworks, cryptographic libraries, web servers, and database engines are predominantly open source, and the available evidence suggests this dominance reflects architectural advantages of community governance, modular design, and the absence of licensing barriers at the scale that modern digital infrastructure requires (Fitzgerald, 2006; Pfandzelter & Bermbach, 2024; Hoffmann et al., 2024). Proprietary software maintains dominance at the interface layer — the user-facing applications, productivity suites, consumer platforms, and managed cloud services through which the overwhelming majority of non-technical users encounter the digital world — and this dominance appears to reflect advantages of centralized quality control, user experience investment, and the ecosystem lock-in that network effects and proprietary data assets create (Wen & Choudhury, 2019; Rikap, 2024). The significance of this asymmetry — open source owns the foundation, proprietary controls the surface — merits emphasis within the scope of this study: As argued by Rikap (2024) and Bowker and Star (1999), the infrastructure layer is technically determinative, whereas the interface layer concentrates economic and political power

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

in the platform-mediated digital economy. Users experience proprietary interfaces; the open source infrastructure is, by definition, invisible to them (Bowker & Star, 1999; Winner, 1980).

A politically honest analysis of the open source versus proprietary debate requires that the distributional question be held continuously in view. As the theoretical framework developed in Section 3 established, every software architecture embeds a distribution of power — over who can inspect, modify, deploy, and benefit from a given technology — and the choice between open and proprietary models is therefore always simultaneously a choice about that distribution (Winner, 1980; Bowker & Star, 1999). The evidence assembled in this article allows this distributional question to be addressed tentatively, within the scope of the cases reviewed. The primary beneficiaries of proprietary software are the vendors who capture economic rents from licensing, the managers who benefit from the accountability structures of vendor support contracts, and the large enterprises whose compliance requirements are more tractable with proprietary certified products (Bonaccorsi & Rossi, 2003; Petcu et al., 2013). The primary beneficiaries of open source software are the organizations that use it without contributing to it — whose effective subsidy from the open source commons represents a transfer of value from the small community of contributing developers to the much larger population of users, whose aggregate demand-side value is estimated at $8.8 trillion against a supply-side production cost of $4.15 billion (Hoffmann et al., 2024). The governance implications of this distribution are significant: The open source ecosystem remains chronically underfunded relative to its economic and social value, as estimated by Hoffmann et al. (2024) and Linåker et al. (2025), and structural responses — the OpenSSF, the Linux Foundation's project funding models, national open source investment programs — represent recognition that the market will not spontaneously correct this underinvestment (Walden, 2020; Linåker et al., 2025).

The third analytical conclusion of this article is normative: organizations, governments, and individual developers cannot afford to treat software model choices as neutral technical decisions. A conscious software policy — one that explicitly evaluates the governance, economic, security, and sovereignty implications of each layer of the technology stack, and that considers the long-term systemic effects of procurement decisions on the digital commons from which all benefit — is an institutional imperative at every level of analysis (Linåker et al., 2025; Bechara & Lechner, 2024). For governments, this means recognizing that procurement of proprietary software for public services creates strategic dependencies that digital sovereignty frameworks are designed to address, and that the principle of public money, public code — software developed with public funds made publicly available — is not merely an ideological preference but a structural mechanism for building

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

institutional capacity and democratic accountability (Neudert, 2024; Santos et al., 2024). Government involvement in open source is increasingly not only pragmatic but politicized, serving to uphold governments' ambitions for national security, international influence, or digital sovereignty, while dilemmas emerge from tensions between securing universally used critical open source components, developing sovereign technologies, and the risk of encroaching on the horizontal and decentralized functioning of open source (Institut français des relations internationales, 2025). For enterprises, conscious software policy means conducting genuine total cost of ownership analyses that account for lock-in costs, ecosystem dependency, and the governance risks of vendor concentration — recognizing that the apparent predictability of proprietary software TCO is bounded by the vendor's unilateral authority over pricing and product continuity (Shaikh & Cornford, 2011; Alhosban et al., 2024). For developers, conscious software policy means understanding that contribution to open source projects is not merely a form of professional development but a civic act that sustains the infrastructure on which the entire digital economy depends — and that the free rider problem documented throughout this article is, in part, a problem of individual professional choices aggregated across millions of practitioners (Lerner & Tirole, 2002; Degefa et al., 2024).

The final reflection of this article is the broadest. The conflict between open source and proprietary software is, at its deepest level, a conflict about who holds the power to define how the digital world functions — who can inspect, modify, govern, and build upon the technological infrastructure that mediates economic exchange, political communication, public administration, scientific research, healthcare, and education at global scale. As highlighted by Pohle (2024) and Rikap (2024), this is not merely a technical question but one of political economy — of whether the foundational infrastructure of the information age is governed as a commons, subject to democratic accountability and collective stewardship, or as a collection of private assets, governed by the commercial interests of the firms that own them and the investors whose financial returns they serve (Pohle, 2024; Glasze et al., 2022; Rikap, 2024).

The evidence reviewed in this article is consistent with the interpretation that neither model, in its pure form, provides a satisfactory answer to this question, though this conclusion is based on the cases and domains examined and may not generalize universally. Proprietary software concentrates governance power in ways that create structural dependencies — commercial, political, and strategic — that undermine the autonomy of users, organizations, and states in ways that are poorly captured by technical capability comparisons or cost analyses. Open source software distributes governance power in ways that are theoretically superior but practically dependent on resource commitments, institutional arrangements, and community health

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

conditions that the open source ecosystem has historically been unable to sustain without external support (O'Mahony, 2007; Curto-Millet et al., 2022). The hybrid configurations that dominate the contemporary landscape — open core, SaaS-on-open-source, corporate-governed open weight AI — combine elements of both without fully resolving the power concentration problem: a corporation can appropriate the community governance benefits of open source during the growth phase and the proprietary control benefits of licensing restriction at commercial maturity, leaving the community that created the value without recourse (West & Gallagher, 2006; Doctorow, 2023; Cooiman, 2024).

The most productive conclusion is therefore not a resolution of the open source versus proprietary debate but a sharpening of the questions it demands. As open source infrastructure becomes ever more deeply embedded in the systems that govern public life, the question of who funds its maintenance becomes inseparable from the question of who controls its governance. As AI systems trained on the world's accumulated knowledge are deployed at large scale, the question of whether their weights, training data, and design decisions are publicly auditable becomes a question about the democratic accountability of some of the most consequential decision-making systems humanity has ever built. As digital sovereignty concerns drive national governments to build indigenous open source infrastructure, the question of who participates in defining the standards of the global digital commons becomes a question about the geopolitical distribution of technological power in the twenty-first century (Pohle, 2024; Widder et al., 2023; Jiang & Belli, 2024). These are questions this article has sought to frame analytically; their resolution requires ongoing empirical research that extends beyond the scope of a single comparative study.

References

ACLU. (2025). Open vs. closed: The battle for the future of language models. American Civil Liberties Union.

Alhosban, A., Kochut, K., & Pesingu, S. (2024). CVL: A cloud vendor lock-in prediction framework. Mathematics, 12(3), 387.

Aliprandi, S., & Muselli, L. (2024). The end of open source? Regulating open source under the Cyber Resilience Act and the new Product Liability Directive. Computer Law & Security Review, 54, 105975.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Anderson, R. (2020). Security engineering: A guide to building dependable distributed systems (3rd ed.). Wiley.

Apache Software Foundation. (2024). Apache Fineract: Financial inclusion for the world. Apache Software Foundation.

Bechara, J., & Lechner, U. (2024). Digital sovereignty and open-source software — A discussion paper. In F. Phillipson et al. (Eds.), Innovations for Community Services. Communications in Computer and Information Science (Vol. 2109, pp. 397–407). Springer.

Belli, L. (2024). Building good digital sovereignty through digital public infrastructures and digital commons in India and Brazil. SSRN Working Paper No. 4966348.

Belton, V., & Stewart, T. J. (2002). *Multiple criteria decision analysis: An integrated approach*. Kluwer Academic Publishers.

Biström, D., Adolfsson, K. K., & Stocchetti, M. (2024). Open-source software and digital sovereignty. In M. E. Auer et al. (Eds.), Smart Technologies for a Sustainable Future. Springer.

Bonaccorsi, A., & Rossi, C. (2003). Why open source software can succeed. Research Policy, 32(7), 1243–1258.

Bowker, G. C., & Star, S. L. (1999). Sorting things out: Classification and its consequences. MIT Press.

Ceruzzi, P. E. (2003). A history of modern computing (2nd ed.). MIT Press.

Chesbrough, H. W. (2003). *Open innovation: The new imperative for creating and profiting from technology*. Harvard Business School Press.

Chowdhury, M., Jawalkar, P., Narasimha, S. N., Sai, A. R., Samsi, S., & Winder, J. (2025). Toward measuring and benchmarking large language model energy and carbon efficiency. arXiv preprint arXiv:2502.16525.

CNCF. (2025). Cloud native 2024: Approaching a decade of code, cloud, and change. Cloud Native Computing Foundation / Linux Foundation Research.

Cong, L. W., He, Z., & Tang, K. (2025). Centralized governance in decentralized organizations. Working Paper, Finance Conference Proceedings.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Connolly, B., Martin, G., & Scanlon, M. (2025). How software design affects energy performance: A systematic literature review. Journal of Software: Evolution and Process, 37(2), e70014.

Contractor, D., McDuff, D., Haines, J. K., Lee, J., Hines, C., Hecht, B., Vincent, N., & Waseem, H. (2023). Behavioral use licensing for responsible AI. *FAccT '23*, 317–328.

Cooiman, F. (2024). Imprinting the economy: The structural power of venture capital. Environment and Planning A: Economy and Space, 56(2), 498–515.

Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2020). A systematic mapping study of software development with GitHub. IEEE Access, 7, 111575–111592.

Cristia, J., Ibarrarán, P., Cueto, S., Santiago, A., & Severín, E. (2017). Technology and child development: Evidence from the one laptop per child program. American Economic Journal: Applied Economics, 9(3), 295–330.

Curto-Millet, D., Haji-Mohammadi, M., & Comyn-Wattiau, I. (2022). The sustainability of open source commons. European Journal of Information Systems, 31(5), 597–613.

Dahlander, L., & Magnusson, M. G. (2008). How do firms make use of open source communities? Long Range Planning, 41(6), 629–649.

Degefa, M. B., Serrano-Solano, B., & Fouilloux, A. (2024). Cultivating community health and well-being in open source: Mitigating burnout and prioritizing mental health. International Journal of Research and Innovation in Social Science, 6(2), 1–8.

Demerouti, E. (2024). Burnout: A comprehensive review. Zeitschrift für Arbeitswissenschaft, 78, 492–504.

Dewi, L. (2009). Total cost of ownership of open source software. Sinergi: Kajian Bisnis dan Manajemen, 11(1), 45–52.

Doctorow, C. (2023). The enshittification of TikTok. Wired.

Durumeric, Z., Kasten, J., Adrian, D., Halderman, J. A., Bailey, M., Li, F., Weaver, N., Amann, J., Beekman, J., Payer, M., & Paxson, V. (2014). The matter of Heartbleed. In Proceedings of the 2014 Conference on Internet Measurement (pp. 475–488). ACM.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Eghbal, N. (2020). *Working in public: The making and maintenance of open source software*. Stripe Press.

Eloul, S., Boutros, F., Ouali, A., & Mukherjee, P. (2024). Toward open banking: Market overview, machine learning, and challenges. IEEE Access, 12, 28411–28437.

Esseola, E., Massey, A., & Singh, R. (2015). Open-source software: Adoption and challenges [Abstract]. ResearchGate. https://www.researchgate.net/publication/277004559

European Commission. (2024). Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union.

European Commission. (2025). Europe's open-source AI landscape: A lever for innovation and sovereignty. [Directorate-General for Communications Networks, Content and Technology].

Fitzgerald, B. (2006). The transformation of open source software. MIS Quarterly, 30(3), 587–598.

Free Software Foundation. (1985/2024). The GNU Manifesto. GNU Project.

Freitag, C., Berners-Lee, M., Widdicks, K., Knowles, B., Blair, G. S., & Friday, A. (2021). The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations. Patterns, 2(9), 100340.

Gangadharan, G. R., Weiss, M., D'Andrea, V., & Iannella, R. (2009). Service license composition and compatibility analysis. In Proceedings of the International Conference on Service Oriented Computing (pp. 296–307). Springer.

Gawer, A., & Cusumano, M. A. (2002). *Platform leadership: How Intel, Microsoft, and Cisco drive industry innovation*. Harvard Business School Press.

Geiger, R. S., Howard, D., & Irani, L. (2021). The labor of maintaining and scaling free and open-source software projects. *Proceedings of the ACM on Human-Computer Interaction, 5*(CSCW1), 1–28.

Georgiou, S., Kechagia, M., Spinellis, D., & Zaidman, A. (2022). Green software: A literature review. Journal of Systems and Software, 189, 111308.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Ghanbari, H., Koskinen, K., & Wei, Y. (2024). From SolarWinds to Kaseya: The rise of supply chain attacks in a digital world. Journal of Information Technology Teaching Cases, 14(2).

Gilbert, R. J., & Rubinfeld, D. L. (2001). Antitrust issues in software markets. In J. Lerner & S. Stern (Eds.), Innovation policy and the economy (Vol. 1, pp. 61–97). MIT Press.

Glasze, G., Cattaruzza, A., Douzet, F., Dammann, F., Bertran, M., Beucher, B., Nicolaci, M., Renard, F., Lacapmesure, A., Mosel, I., Münkler, L., & Schulze, P. (2022). Contested spatialities of digital sovereignty. Geopolitics, 28(2), 417–439.

Gliściński, K. (2025). Intellectual property rights as private rights: Implications of the theory of internally limited rights and incentive theory. The Journal of World Intellectual Property, 28(1), 1–25.

Grad, B. (2002). A personal recollection: IBM's unbundling of software and services. IEEE Annals of the History of Computing, 24(1), 64–71.

Hazlett, T. W. (2000). Microsoft's Internet exploration: Predatory or competitive? Journal of Law and Politics, 29(4), 327–392.

Henderson, P., Mitchell, E., Manning, C., Jurafsky, D., & Liang, P. (2023). Foundation models and fair use. *arXiv preprint arXiv:2303.15715*.

Hoffmann, M., Nagle, F., & Zhou, Y. (2024). The value of open source software. Harvard Business School Strategy Unit Working Paper No. 24-038.

Hong, S., & Ji, S. (2024). Political determinants of government transparency: Evidence from open government data initiatives. Politics & Policy, 52(3).

Huang, K.-F., Dyerson, R., Wu, L.-Y., & Harindranath, G. (2013). From temporary competitive advantage to sustainable competitive advantage. British Journal of Management, 26(1), 1–17.

IBM. (2019). IBM closes landmark acquisition of Red Hat for $34 billion. IBM Newsroom.

Institut français des relations internationales. (2025). Software power: The economic and geopolitical implications of open source software. IFRI.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Institute for Global Change. (2026). Sovereignty in the age of AI: Strategic choices, structural dependencies and the long game ahead. Tony Blair Institute for Global Change.

ITIF. (2025). Tip of the iceberg: Understanding the full depth of Big Tech's contribution to US innovation and competitiveness. Information Technology and Innovation Foundation.

Izaiku, Q., Serebrenik, A., & Vasilescu, B. (2023). Diversity in DevOps practices in open source. In Proceedings of the 2023 IEEE/ACM International Conference on Mining Software Repositories (pp. 1–12). IEEE.

Jansen, S., Bloemendal, E., & Piotrowski, M. (2021). Governance challenges for blockchain and decentralized autonomous organizations. Computers in Industry, 129, 103457.

Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). A sense of community: A research agenda for software ecosystems. In Proceedings of the 31st International Conference on Software Engineering (pp. 187–190). IEEE.

Jawhari, B., Ludwick, D., Keenan, L., Zakus, D., & Hayward, R. (2016). Benefits and challenges of EMR implementations in low resource settings: A state-of-the-art review. BMC Health Services Research, 16, 116.

Jiang, M., & Belli, L. (Eds.). (2024). Digital sovereignty in the BRICS countries: How the Global South and emerging power alliances are reshaping digital governance. Cambridge University Press.

Jiang, M., Zhao, J., & colleagues. (2024). Models of state digital sovereignty from the Global South. Policy & Internet, 16(4).

Kapoor, S., & Narayanan, A. (2023). Leaky pipeline: How open-source LLMs fall short of reproducibility. *arXiv preprint arXiv:2311.06523*.

Kodhek, G. A. (2024). A model for total cost determination in open-source software ownership: Case of Kenyan universities' learning management system. Computer and Information Science, 17(1).

Lakhani, K. R., & von Hippel, E. (2003). How open source software works: "Free" user-to-user assistance. Research Policy, 32(7), 923–943.

Lemenager, M. (2022). Moodle: The open source LMS. LisTedTECH Research.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Lenk, A., Klems, M., Nimis, J., Tai, S., & Sandholm, T. (2011). What's inside the cloud? An architectural map of the cloud landscape. In Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing (pp. 23–31). IEEE.

Lerner, J., & Tirole, J. (2002). Some simple economics of open source. Journal of Industrial Economics, 50(2), 197–234.

Liesenfeld, A., & Alikhani, M. (2024). Rethinking open source generative AI: Open-washing and the EU AI Act. *TheWebConf '24 Companion*, 1029–1036.

Linåker, J., Papatheocharous, E., & Olsson, T. (2025). Advancing digital government: Integrating open source software enablement indicators in maturity indexes. arXiv preprint arXiv:2510.04603.

Linnenluecke, M. K., & colleagues. (2025). The five stages of the enshittification of academic publishing. Organization.

Linthicum, D. (2016). Critical analysis of vendor lock-in and its impact on cloud computing migration: A business perspective. Journal of Cloud Computing, 5(1), 4.

Maffie, M. D., & Hurtado, D. A. (2025). Enshittification and gig-economy labor conditions. British Journal of Industrial Relations.

Massacci, F., Jaeger, T., & Peisert, S. (2021). SolarWinds and the challenges of patching: Can we ever stop dancing with the devil? IEEE Security & Privacy, 19(2), 14–19.

Mayrhofer, R., Stoep, J. V., Brubaker, C., & Kralevich, N. (2021). The Android platform security model. ACM Transactions on Privacy and Security, 24(3), Article 19.

Meta AI. (2024). Introducing Llama 3.1: Our most capable models to date. Meta AI Research.

Mikolajczak, C., & Goanta, C. (2023). Achieving a (copy)right to repair for the EU's green economy. Journal of Intellectual Property Law & Practice, 18(5), 344–357.

Mikolajczak, C., & Trogal, K. (2024). On "the politics of repair beyond repair": Radical democracy and the right to repair movement. Journal of Business Ethics, 194(2), 377–393.

JIRRS
Journal International Review of Research Studies

Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Mökander, J., Schuett, J., Kirk, H. R., & Floridi, L. (2023). Auditing large language models: A three-layered approach. *AI and Ethics, 3*, 1085–1115.

Momani, A. M. (2010). Comparison between two learning management systems: Moodle and Blackboard. SSRN Working Paper.

Mügge, D. (2024). EU AI sovereignty: For whom, to what end, and to whose benefit? Journal of European Public Policy, 31(8), 2200–2225.

Nagle, F., Greenstein, S., & Wright, N. L. (2024). Open source software and entrepreneurial growth. Strategic Management Journal, 45(9).

Neudert, L.-M. (2024). Reclaiming digital sovereignty: Policy and power dynamics behind Germany's NetzDG. Journal of Information Policy, 14, 417–470.

O'Mahony, S. (2007). The governance of open source initiatives. [Journal of Management & Governance, 11(2), 139–150].

Olivier, J. (2021). The use of open educational resources during COVID-19: Navigating a 'new normal.' Research in Social Sciences and Technology, 6(2), 143–158.

Open Future. (2025). AI Act and open source: Observatory archive. Open Future Foundation.

OpenSSF. (2024). XZ backdoor CVE-2024-3094. Open Source Security Foundation.

OpenTofu. (2023). OpenTofu manifesto. Linux Foundation.

Orlikowski, W. J. (1992). The duality of technology: Rethinking the concept of technology in organizations. *Organization Science, 3*(3), 398–427.

OSI. (2024). The open source AI definition — Version 1.0. Open Source Initiative.

Ostrom, E. (1990). Governing the commons: The evolution of institutions for collective action. Cambridge University Press.

Perens, B. (1999). The open source definition. In C. DiBona, S. Ockman, & M. Stone (Eds.), Open sources: Voices from the open source revolution (pp. 171–188). O'Reilly Media.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Petcu, D., Vasilakos, A. V., & Rak, M. (2013). Portability in clouds: Approaches and research opportunities. Scalable Computing: Practice and Experience, 14(3), 219–233.

Pfandzelter, T., & Bermbach, D. (2024). Will serverless end the dominance of Linux in the cloud? In Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS '24). ACM.

Pirozzi, A., Visaggio, C. A., & Ferraro, A. (2025). Wolves in the repository: A software engineering analysis of the XZ Utils supply chain attack. arXiv preprint arXiv:2504.17473.

Pohle, J. (2024). Unthinking digital sovereignty: A critical reflection on origins, objectives, and practices. Policy & Internet, 16(1), 1–20.

Ragnedda, M., & Muschert, G. W. (Eds.). (2013). The digital divide: The internet and social inequality in international perspective. Routledge.

Raman, N., Cao, M., Tourani, P., Vasilescu, B., & Kästner, C. (2020). Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results (pp. 81–84). ACM.

Raymond, E. S. (1999). The cathedral and the bazaar. Knowledge, Technology & Policy, 12(3), 23–49.

Riehle, D. (2009). The commercial open source business model. In Value Creation in e-Business Management (pp. 18–30). Springer.

Rikap, C. (2024). Varieties of corporate innovation systems and their interplay with global and national systems: Amazon, Facebook, Google and Microsoft's strategies to produce and appropriate artificial intelligence. Review of International Political Economy, 31(6), 1735–1763.

Rozas, D., Tenorio-Fornés, A., Díaz-Molina, S., & Hassan, S. (2021). When Ostrom meets blockchain: Exploring the potentials of blockchain for the governance of the commons. Frontiers in Blockchain, 4, 577680.

Santos, C. M., da Silva, H. L., & Cappelli, C. (2024). Open government data in the Brazilian digital government: Enabling an SDG acceleration agenda. Government Information Quarterly, 41(4), 101970.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Schweik, C. M., & Kitsing, M. (2010). Applying Elinor Ostrom's rule classification framework to the analysis of open source software commons. Transnational Corporations Review, 2(1), 13–26.

Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. Management Science, 52(7), 1000–1014.

Shaikh, M., & Cornford, T. (2011). Total cost of ownership of open source software: A report for the UK Cabinet Office. London School of Economics Enterprise / OpenForum Europe.

Shaikh, M., Vayani, A. H. M., Akram, S., & Qamar, N. (2022). Open-source electronic health record systems: A systematic review of most recent advances. Health Informatics Journal, 28(2).

Shrivastava, R., & Korkmaz, G. (2024). Measuring public open-source software in the federal government: An analysis of Code.gov. Journal of Data Science, 22(3), 356–375.

Solaiman, I. (2023). The gradient of generative AI release: Methods and considerations. *FAccT '23*, 111–122.

Srivastava, A., & Mishra, S. (2025). Digital sovereignty and AI: Developing India's National AI Stack for strategic autonomy. Procedia Computer Science, 253, 1–10.

Stallman, R. (1985). The GNU Manifesto. Dr. Dobb's Journal, 10(3), 30–34.

Star, S. L., & Ruhleder, K. (1996). Steps toward an ecology of infrastructure. Information Systems Research, 7(1), 111–134.

Stiglitz, J. E. (2008). Economic foundations of intellectual property rights. Duke Law Journal, 57(6), 1693–1724.

Stodden, V., McNutt, M., Bailey, D. H., Deelman, E., Gil, Y., Hanson, B., Heroux, M. A., Ioannidis, J. P. A., & Taufer, M. (2016). Enhancing reproducibility for computational methods. Science, 354(6317), 1240–1241.

Stol, K.-J., & Fitzgerald, B. (2015). Inner source — Adopting open source development practices in organizations: A tutorial. IEEE Software, 32(4), 60–67.

Synopsys. (2024). Open source security and risk analysis (OSSRA) report 2024. Synopsys/Black Duck.

JIRRS
Journal International Review of Research Studies
Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Tai, K.-T. (2021). Open government research over a decade: A systematic review. Government Information Quarterly, 38(2), 101571.

Terrell, J., Kofink, A., Middleton, J., Rainear, C., Murphy-Hill, E., Parnin, C., & Stallings, J. (2017). Gender differences and bias in open source: Pull request acceptance of women versus men. PeerJ Computer Science, 3, e111.

Tolu, H. (2018). Free software philosophy and open source. International Journal of Open Source Software and Processes, 10(1), [1–18].

Trinkenreich, B., Guizani, M., Wiese, I., Tamburri, D. A., & Gerosa, M. (2023). Pots of gold at the end of the rainbow? Predictors of open source developer burnout. *IEEE Transactions on Software Engineering, 49*(4), 2064–2079.

Trinkenreich, B., Guizani, M., Wiese, I., Sarma, A., & Gerosa, M. (2021). Women's participation in open source software: A survey of the literature. ACM Transactions on Software Engineering and Methodology, 31(1), Article 7.

Trist, E. L., & Bamforth, K. W. (1951). Some social and psychological consequences of the longwall method of coal-getting. *Human Relations, 4*(1), 3–38.

Tsay, J., Dabbish, L., & Herbsleb, J. (2014). Influence of social and technical factors for evaluating contribution in GitHub. In Proceedings of the 36th International Conference on Software Engineering (pp. 356–366). ACM.

UNESCO. (2019). Recommendation on open educational resources (OER). United Nations Educational, Scientific and Cultural Organization.

UNIDO. (2003). The role of intellectual property rights in technology transfer and economic growth. United Nations Industrial Development Organization.

U.S. Department of Commerce, Bureau of Industry and Security. (2024). Kaspersky Lab, Inc. prohibition: Final determination. Office of Information and Communications Technology and Services.

van Dijk, J. (2020). The digital divide. Polity Press.

Ven, K., & Verelst, J. (2008). The impact of ideology on the organizational adoption of open source software. Journal of Database Management, 19(2), 58–72.

Walden, J. (2020). The impact of a major security event on an open source project: The case of OpenSSL. In Proceedings of the 17th International Conference on Mining Software Repositories (pp. 32–42). ACM.

JIRRS
Journal International Review of Research Studies

Received: 10/02/2026 - **Accepted:** 18/03/2026
Vol: 01.03
DOI: 10.66104/hnyd5f72
Pages: 1-98

Warschauer, M., & Ames, M. (2010). Can one laptop per child save the world's poor? Journal of International Affairs, 64(1), 33–51.

Weiss, M., Muschick, D., & Boettcher, M. (2022). Archetypes of open-source business models. Electronic Markets, 32(3), 1449–1464.

Wen, W., & Choudhury, V. (2019). Open source adoption strategy. Electronic Commerce Research and Applications, 36, 100869.

West, J., & Gallagher, S. (2006). Challenges of open innovation: The paradox of firm investment in open-source software. R&D Management, 36(3), 319–331.

Widder, D. G., West, S., & Whittaker, M. (2023). Open (for business): Big tech, concentrated power, and the political economy of open AI. AI Now Institute.

Williams, L., & Zacchiroli, S. (2025). Research directions in software supply chain security. ACM Transactions on Software Engineering and Methodology, 34(3), Article 64.

Winner, L. (1980). Do artifacts have politics? Daedalus, 109(1), 121–136.

Xie, C., & Hu, S. (2024). Open banking: An early review. Journal of Internet and Digital Economics, 4(1).

XtendedView. (2025). Linux statistics 2026: What's surging and why. https://xtendedview.com/linux-statistics/

Zaggl, M. A., Rees, J., & Sunyaev, A. (2025). Certification of open source software compliance: Insights from a conjoint experiment. Information Systems Journal, 35(1).

Zanjani, N. (2022). A systematic review on trends in using Moodle for teaching and learning. International Journal of STEM Education, 9(1), Article 9.

Zeber, D., Amann, J., Vallentin, M., Sommer, R., & Weaver, N. (2015). Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. Communications of the ACM, 58(7), 48–56.