2003

How open is open enough?: Melding proprietary and open source platform strategies

Joel West
San Jose State University, joel.west@sjsu.edu

Recommended Citation
Joel West. "How open is open enough?: Melding proprietary and open source platform strategies" *Research Policy* (2003): 1259-1285. doi:10.1016/S0048-7333(03)00052-0

How Open is Open Enough?
Melding Proprietary and Open Source Platform Strategies

Joel West[1]
College of Business, San José State University, One Washington Square, San José, CA 95192-0070 USA

December 31, 2002

Forthcoming in *Research Policy* special issue on "Open Source Software Development"
(Eric von Hippel and Georg von Krogh, editors)

Abstract

Computer platforms provide an integrated architecture of hardware and software standards as a basis for developing complementary assets. The most successful platforms were owned by proprietary sponsors that controlled platform evolution and appropriated associated rewards.

Responding to the Internet and open source systems, three traditional vendors of proprietary platforms experimented with hybrid strategies which attempted to combine the advantages of open source software while retaining control and differentiation. Such hybrid standards strategies reflect the competing imperatives for adoption and appropriability, and suggest the conditions under which such strategies may be preferable to either the purely open or purely proprietary alternatives.

Keywords: open source, standards competition, computer architecture, innovation returns

Acknowledgments

I would like to thank Jason Dedrick, Tineke Egyedi, Scott Ensign and two anonymous reviewers for helpful suggestions and comments, as well as seminar participants at Case Western's Weatherhead School of Management and the 2002 Academy of Management annual meeting. I am especially grateful to the special issue editors for advice and encouragement that were instrumental in developing the paper.

My appreciation goes to the representatives of Apple, IBM and Sun who generously shared their time to both explain various complex technologies and also the institutional relationships behind those technologies.

The remaining errors are of course my own.

[1] Tel: +1-408-924-7069; fax: +1-408-924-3555. *E-mail address:* Joel.West@sjsu.edu

1. Introduction

The evolution of the computer industry has been driven by the emergence of standardized platforms which allow modular substitution of complementary assets such as software and peripheral hardware.

The initial platforms were proprietary, in which a computer systems manufacturer controlled all hardware and software layers of the standards architecture. These platforms were later challenged by two hardware-independent operating systems — Unix and Windows — which reduced differentiation between hardware vendors and shifted platform control to the operating system vendors. The Unix operating system also inspired a more radical shift, the open source movement in which Linux allowed users and competitors to control a platform's direction.

These various strategies reflect the essential tension of *de facto* standards creation: that between appropriability and adoption. To recoup the costs of developing a platform, its sponsor must be able to appropriate for itself some portion of the economic benefits of that platform. But to obtain any returns at all, the sponsor must get the platform adopted, which requires sharing the economic returns with buyers and other members of the value chain.

The proprietary and open source strategies correspond to the two extremes of this trade-off. In making a platform strategy for the 21st century, leading computer vendors face a dilemma of how much is open enough to attract enough buyers while retaining adequate returns.

First I review the theory and history of proprietary computer platform strategies, contrasting that with the Unix-based open systems movement. I then examine the origins and motivations of Linux and other open source software projects. In three abbreviated case studies, I present the hybrid strategies of three platform vendors — Apple Computer, IBM and Sun Microsystems — that combine open source and proprietary platform strategies in hopes of obtaining competitive advantage. Finally an analysis of these cases is combined with prior research to suggest the theoretical implications of such hybrid strategies.

2. Proprietary Platform Strategies

2.1 Dynamics of Proprietary Platform Competition

A proprietary platform consists of an architecture of related standards, controlled by one or more sponsoring firms. For a computer system, the architectural standards typically encompass a processor, operating system (OS), and associated peripherals. Some have also extended the concept of a "platform" to include multiple layers of software, such as applications that rely on a "middle ware" tool such as Java or a database (Morris and Ferguson, 1993; Bresnahan and Greenstein, 1999; West and Dedrick, 2000).

A platform is but a specific example of the general class of technological innovations studied by Teece (1986), who links the ability of firms to profit from their technological innovations to the appropriability regime for intellectual property rights (IPR) — either through formal *de jure* protection (e.g. patents) or through *de facto* protection such as tacit knowledge or trade secrets. Absent such IPR protection, firms selling a given technology can be expected to adopt marginal cost pricing and drive profit margins to zero (Katz and Shapiro, 1986; Beggs and Klemperer, 1992). Without appropriability, Teece (1986) suggests that firms must use some combination of speed, timing and luck if they hope to appropriate returns generated by their innovation.

Teece (1986) also considers those innovations (such as computer systems) that require the provision of complementary assets to commercialize the innovation. When additional investment is required to co-specialize the asset to be useful with a given innovation, the successful adoption of the innovation and the related assets are mutually reinforcing, providing a positive feedback cycle. Thus, to make a successful "whole product" solution, the owner of the innovation seeks to attract such complementary assets, in part by sharing the overall returns of the innovation with the

third party supplier of such assets (Katz and Shapiro, 1985; Teece, 1986; Moore, 1991).

The positive feedback, self-reinforcing cycle of success between a *de facto* standard and its co-specialized asset success has been termed "network externalities" (Katz and Shapiro, 1985) or "demand side economies of scale" (Katz and Shapiro, 1986). When such network effects are coupled with switching costs between standards and high up front R&D costs, Arthur (1996) predicts that the dominant technology will enjoy "increasing returns to scale" that magnify an early lead in a technology contest. The instability and self-reinforcing nature of such a lead has often been referred to as "tipping" of the contest (e.g. Besen and Farrell, 1994). Liebowitz and Margolis (1999) argue that actual tipping is rare, and that lasting success is more often explained by production economies of scale and firm execution.

In the case of computing platforms, most research has focused on one particular type of complementary asset, that of prepackaged application software. A computer platform is not, in itself, useful without software to solve specific problems. During the 1960s and 1970s, large organizations buying mainframe computers typically developed their own custom software. However, the advent of the mass-market personal computers attracted many new users unable to develop their own software, fueling a shift to prepackaged application software packages (Mowery, 1996). As such, the control of the platform's complementary assets (e.g. packaged software) is determined by the ability to create and evolve application programming interfaces (APIs), which specify how application software must be co-specialized to work with a particular platform (West and Dedrick, 2000).

Multiple platforms can and have simultaneously co-existed serving different market segments (Table 1). Bresnahan and Greenstein (1999) argue that in the U.S. computer industry, new platforms succeeded when they tapped unserved market niches, avoiding competition with established platforms until they achieved critical mass. Due to simple economies of scale, mass market platforms displaced more specialized products, either by providing lower cost or addressing a broader range of buyer needs (Morris and Ferguson, 1993).

Firms that successfully establish and maintain a proprietary platform enjoy the right to appropriate the returns from a success of that platform (Morris and Ferguson, 1993). But platform success is a necessary but not sufficient condition for profiting from proprietary technology innovation. When competing firms control different layers of the standards architecture, platform leadership is unstable because control of the platform can shift without disrupting the buyer's value proposition. In particular, a firm that can take control of access to the complementary assets has an incentive to do so to capture the returns from the platform (Bresnahan and Greenstein, 1999; West and Dedrick, 2000).

In explaining platform success, economic research has focused on demand and supply side economies of scale, and the broad strategic choices made by sponsoring firms. However, there is also repeated evidence that operational execution is crucial to the relative success or failure of individual platforms (Morris and Ferguson, 1993; Liebowitz and Margolis, 1999; West, 2003). Also, measures of platform success have focused on adoption or market share. The more managerially relevant metric would be the sponsor's net profit or return on investment from the proprietary technological innovation, although such data is much harder for researchers to obtain.

2.2 Mainframes: Vertically Integrated Proprietary Platforms

In 1964, IBM introduced the world's first successful computing platform, the System/360. A key success factor was the modular architecture that enabled the use of the same software and peripherals throughout the product line, providing interoperability that was missing the product lines of IBM and other companies. IBM also leveraged its existing domestic market share and global reach to win the largest share of the global market, vanquishing the proprietary platforms of domestic rivals and European national champions (Flamm, 1988; Chandler, 1997; Moschella, 1997).

The one exception to IBM's dominance was in Japan, where by the 1990s IBM held only a 25% share as part of a stable, four firm oligopoly, in which IBM's rivals produced clones of the S/360 platform (Anchordoguy, 1989; Ferguson and Morris, 1993).

IBM was a vertically integrated manufacturer of processors, systems, peripherals and software for the S/360 platform and its System 370 and System 390 successors. The pattern was repeated by Digital Equipment Corporation, which introduced a series of minicomputer platforms, culminating with its most successful offering, the VAX 32-bit minicomputer (1978) with its proprietary VAX-11 processor and VAX/VMS operating system. DEC dominated U.S. technical markets, although IBM led business markets with its AS/400 minicomputer (Bresnahan and Greenstein, 1999).

Late entrants had trouble gaining market share for their innovations due to switching costs between the proprietary platforms (Greenstein, 1997). A major cost in converting from one platform to another was due to application software, which had to be specialized to fit each platform's APIs. In the 1960s and 1970s, this tended to be software custom-developed to meet the firm's particular needs, but subsequent buyer shifts to using off-the-shelf packaged software shifted the conversion cost from users to third-party software suppliers.

Chandler (1997) argues that the concentration of the global mainframe industry was consistent with the pattern of other capital-intensive industries, in that the high entry costs allow the pioneer and early challengers to form a stable oligopoly. Bresnahan and Greenstein (1999) attribute such oligopoly to the effect of endogenous sunk costs in rewarding scale; their mechanisms and predicted outcomes correspond to Arthur's (1996) formulation of increasing returns to scale.[2]

[2] The Bresnahan and Greenstein formulation also identifies platform-specific investments in providing market stability, corresponding to Teece's (1986) earlier analysis of specialized complementary assets.

2.3 Personal Computer Brings Horizontal Platform Control

As with mainframes and minicomputers, the personal computer industry attracted many new entrants that tried and failed to establish successful platforms. However, there was one crucial difference: the 1971 invention of the microprocessor dramatically lowered the cost of entry and also led to platform convergence as a large number of systems makers purchased processors from a shrinking number of microprocessor vendors. During the initial 8-bit era, the PC industry used two different platform strategies. One group used the Intel 8080 (or compatible) processors along with CP/M, a proprietary operating system licensed to many computer makers. The other group bought an inexpensive processor and then designed their own software to run on top of it. Application software was either designed for CP/M APIs, or for one of the proprietary platforms.

Seeing the growth of the PC market, and worried about ceding market control to early pioneers, in 1980 IBM launched a crash project to build a 16-bit PC. IBM's mainframe power and reputation assured the success of the IBM PC, which also rendered 8-bit PCs obsolete. The IBM PC standard soon dominated the world, except in Japan where NEC's proprietary PC-98 dominated the market from 1983-1995 (Chposky and Leonsis, 1988; West and Dedrick, 2000).

As with CP/M, IBM's PC architecture used both a processor and operating system from outside vendors. When coupled with its unexpected legal defeats on ROM copyrights, IBM lost control of its platform as other firms produced "clone" computers that ran the same application software (Langlois, 1992). For the next decade, IBM spent billions of dollars on proprietary technologies in an unsuccessful attempt to re-assert its leadership of the PC industry.

Instead, the "IBM PC" platform gradually began to be termed the "Wintel" platform, an acknowledgment that the proprietary platform control rested with Microsoft (Windows) and Intel. Both firms enjoyed the barriers to imitation provided by economies of scale for R&D and network effects through software supply (Arthur,

1996). Grove (1996) argues that the resultant horizontal specialization of the PC industry is more efficient (and thus more durable) than the vertically integrated structure, because it allows for the producer of each layer to achieve economies of scale by serving the broadest possible market.

2.4 Workstations: Unix and Open Systems

The horizontally specialized platform strategy predates CP/M and MS-DOS, but instead began with AT&T's Unix operating system. Unix began in 1969 with minicomputers, helped create the new workstation platforms of the 1980s, and later became an important multivendor, multi-product mainframe platform. Unix eventually evolved into a new form of open non-proprietary platform standard, often referred to as the "open systems" movement (Gabel, 1987).

As with any other operating system, Unix was not a complete platform specification, because each hardware system might have different processor and peripheral interface standards (Table 2). However, Unix quickly evolved into a portable OS that tended to "hide" the differences between hardware from software applications, and so could present a set of common APIs across widely divergent hardware implementations. The task was aided by the C programming language which served as a highly efficient substitute for hardware-dependent assembly language.

AT&T was restricted from selling computer products by a 1956 anti-trust settlement, so in the first 15 years Unix was limited to internal use, research universities, and a comparatively small number of user companies and hardware vendors that bought source code licenses. AT&T eventually spun off Unix into a separate company, which was sold to Novell and later SCO. In the 1980s, Unix became the preferred operating system for computer workstations and also won significant market share in minicomputers, high end computer servers and supercomputers.

In some ways, AT&T's Unix strategies in the 1980s paralleled those of Microsoft with MS-DOS and later Windows. By licensing their operating systems to multiple hardware vendors, each made their platform ubiquitous by reducing switching costs and differentiation between hardware vendors. Both operating systems shared APIs across multiple hardware vendors. As with even the most proprietary computing platform, UNIX and MS-DOS were "open" to third party software suppliers, utilizing APIs widely disseminated to maximize software availability.

Unix and Windows were also similar in that the shared OS nearly eliminated the antecedents of Arthur's (1996) typology — economies of scale, network effects and switching costs — that might lead to positive returns to scale, and thus tip the market share contest to a single winner among computer manufacturers.[3] The lack of such factors made it unlikely that any manufacturer would enjoy market dominance. In Unix workstations, most of the market was fragmented among four major firms — Sun, IBM, HP and DEC (later Compaq) — none of whom captured more than a 35% market share.

However, in the 1990s Unix and Windows differed dramatically in the control of their APIs. Windows retained its proprietary APIs under the control of a single firm which produced the only implementation, allowing it to enjoy monopoly rents. Meanwhile, *de facto* control of the Unix APIs had shifted to various industry committees and consortia in the "open systems movement," which published vendor-independent standards such as POSIX, OSF/1 and X/OPEN. While the trademarked "Unix" was all derived from AT&T's IPR, the "open systems" evolved into multiple independent implementations

---

3 For both Unix and Windows machines, the computer makers spent little on operating system research and development, eliminating that as a source of economies of scale. The shared APIs also provided access to the same complementary assets and reduced switching costs between computer makers. Between "Wintel" machines, the complementary assets were identical and switching costs negligible; for Unix workstations, the processor differences and proprietary API extensions slightly increased the switching costs for users and suppliers of application software, but were much lower than between proprietary platforms.

— including several "open source" implementations — each compliant with the accepted POSIX specification. Thus, the open systems movement reflected an evolution of platform strategies that reduced the ability of any individual firm to obtain control or differentiation for their platform (Table 3). However, the subsequent "open source" movement took this to the next level.

2.5 Assessment

After IBM succeeded with its System/360 platform, rival computer makers sought to emulate IBM's vertically integrated proprietary platform strategy. Positive returns to scale meant the winning proprietary platform enjoyed high barriers to imitation and thus high profits. Vertical integration allowed a firm to appropriate those profits without having to share them with other firms. But when a single platform (like the IBM S/360) enjoyed sustained market share dominance, rivals had trouble competing with a vertically integrated proprietary strategy: with a smaller share, they lacked minimum efficient scale to cover the fixed R&D costs.

As an interim measure, firms procured non-critical components (such as memory and peripherals) from common suppliers, developing a proprietary processor and operating system to provide differentiation. However, such component sourcing did not address the processor and OS R&D costs, and still left market share laggards at a significant disadvantage due to fewer complementary assets (typically software) and switching costs faced by most potential users.

So over the longer term, minor computer makers tried various strategies to pool R&D and the supply of complementary assets across multiple producers; these strategies also allowed firms without proprietary platforms to enter new markets. One such strategy was to buy the crucial OS or processor from external proprietary vendors (such as AT&T, Microsoft or Intel). Another was to form a multi-vendor consortia (e.g. the Open Software Foundation) to pool technology among vendors. By sharing some (or all) of their platform, they enjoyed better adoption but risked intra-platform competition that limited their ability to profit from the platform success, even for firms (like IBM) that had been the platform's original innovator.

While these newer platform strategies gave computer users lower switching costs and higher bargaining power, buyers still fundamentally licensed technology that was owned by computer vendors (or their alliances) — owners that could set the terms and pricing of the technology, as well as the schedule for enhancements and error correction.[4] This relationship fundamentally changed with the emergence of "open source" computing platforms.

3. Emergence of Open Source Platforms

In developed countries, software enjoys strong intellectual property rights (IPR) protections in the form of trade secrets, copyrights and (most recently) patents. The ability to create and modify software products is governed by the access to the source code, which is why for-profit software developers have historically treated such source code as a closely guarded trade secret.

So-called "open source" software represents the antitheses of a proprietary technology strategy. Rather than using formal IPR protection to set boundaries between vendors and their competitors and customers, open source enlists all as collaborators, maximizing adoption throughout the value chain but minimizing the options for appropriating rents from the software.

The success of the open source movement reflected a confluence of three factors in the mid 1990s (West and Dedrick, 2001):

- users seeking an inexpensive Unix implementation free of AT&T licensing restrictions;

[4] The open systems standards-setting consortia included representatives of major computer buyers, but the deployment of new technologies implementing these standards remained under the control of the computer makers.

- a philosophical movement rejecting the idea of software ownership and appropriability;
- emergence of the Internet as both an enabler and objective for collaborative software development.

3.1 Linux and Other Unix-like Platforms

In the 1980s, the Unix platform had three main attractions for programmers: it ran on inexpensive minicomputers, was hardware independent and provided a state of the art environment for software development. Thus, it was a natural target for those who eventually developed "clone" operating systems.

In 1984, Richard Stallman left his job as an MIT programmer to develop a free Unix-like operating system. He founded Project GNU, which by 1990 had produced a variety of software development tools, but lacked the essential core of a modern operating system, a kernel (Stallman, 1999). Efforts to develop a kernel had first been delayed by work on other components, and then floundered for several years without strong leadership.

Two groups used the GNU tools as the basis for assembling complete and free Unix "clone" operating systems. In 1991, Linus Torvalds began writing a UNIX-compatible operating system for his new PC, and solicited others to join in his efforts. By early 1993, a version of Linux was freely available for downloading on the Internet (Varhol, 1994; Torvalds and Diamond, 2001).

Meanwhile, from 1990-1992 a Unix team at the University of California, Berkeley solicited outsiders to volunteer to rewrite components for its BSD Unix using only the published APIs (and thus not violating the copyright and trade secrets of AT&T's source code). This in turn spawned a series of BSD implementations primarily aimed at PC hardware: NetBSD (1993), FreeBSD (1993) and OpenBSD (1996) (McKusick, 1999; West and Dedrick, 2001).

By relying heavily upon the Project GNU components, both the Linux and various BSD teams delivered free Unix-compatible operating systems originally oriented at personal computer hobbyists. Through cost and flexibility advantages, they gradually supplanted other Unix distributions for Intel-based personal computers, and also enabled such PCs to be used as reliable servers by organizations. Despite contrasting strategies for control and IPR, both the Linux and BSD groups became forerunners of what later was named the "open source" movement.

3.2 "Free Software" vs. "Open Source"

In his academic computing career, Stallman came to expect a computing environment where users shared software and could make custom modifications (Stallman, 2001). Such sharing had also existed between some large computer sites, through computer user groups like SHARE (for IBM users) and DECUS (for Digital Equipment Corp. users), and in a few publicly distributed software programs such as the TENEX operating system and the sendmail mail server. In launching Project GNU, Stallman promoted a philosophy diametrically opposed to the norms of proprietary commercial software. Extending by analogy the traditions of pooled scientific research and the free dissemination of ideas, Stallman argued that all software should be "free software," with source code that can be read, modified and redistributed (Zachary, 1991; Stallman, 1999).

Some but not all of Stallman's goals were shared by other developers of free software. By 1998, Linux and other technologies had become popular but Stallman's ideology had won only limited commercial support. To promote adoption by business users and third-party developers, firms that sold Linux- and GNU-related support and services met to promote a more business-friendly concept of collaborative software development. They labeled their common vision "open source" (DiBona et al., 1999: 3).

Unlike for proprietary software, for both "open source" and "free software" the source code and executable component are freely distributed. Both can be freely downloaded from the Internet, even though some users decide to buy a distribution on tape or CD-ROM. And in both cases, consulting, support and training services can and are sold without restriction.

The major difference is that "free software" prohibits *ex post* appropriation of the technology: any derivative works must also be distributed as "free software" and all changes returned to the original author for subsequent redistribution. Stallman (1999) argued that this is essential to prevent firms from making minor improvements to free software and then using it as a way to attract users to their non-free commercial upgrade. The Project GNU tools and the Linux OS were distributed under the restrictions of the so-called GNU Public License (GPL).

Meanwhile, "open source" projects did not impose any such restrictions, allowing individuals or firms to customize and combine open source software as they desired; however, the profit potential of minor improvements is limited by the availability of the free alternative. This form of source code license was developed by the BSD Unix clones and the Apache web server, and was later emulated by other projects that used "BSD-style" or "Apache-style" licenses (West and Dedrick, 2001).[5]

The two licenses differ in their competitive implications. A developer releasing source code with a BSD-style license grants the most rights, in that others (including competitors) can modify and use the software as they please. Using the GPL levels the playing field: all users are required to share any subsequent changes, eliminating the ability of any party (including the original owner and any competitors) to differentiate their offerings through software enhancements.

They are similar in that for both licenses, a software developer voluntarily surrenders the ability to appropriate the returns from its R&D in hopes of winning greater adoption. Although they cannot directly profit from such software, developers could use free software as a complementary asset to help sell hardware (or other software), or could use it to sell assets complementary to the free software — such as consulting, support and training.

[5] Henceforward, I use "Open Source" to subsume both the BSD/Apache-style licenses and GNU-style "free software" licenses.

4. Context for the Study

This study examines the decision by proprietary platform vendors to release and support open source technologies as part of their platform strategies in the period 1995-2002 (Table 4). The field study and analysis of secondary data were intended to explain this somewhat paradoxical development. Was it a temporary phenomenon tied to a particular time and place, e.g. the importance of supporting Internet standards? Was it a desperation phenomenon for firms that were eventually destined to fail — as happened during the early 1990s when failing minicomputer makers abandoned their proprietary OS to adopt open systems? Or was this a sustainable and ongoing business model for makers of computer hardware and integrated computing platforms?

The study considers three companies — Apple, IBM and Sun — that historically had promoted vertically integrated platforms differentiated by proprietary software in (respectively) the PC, mainframe and workstation eras. As these proprietary strategies began to falter, each eventually employed open source to revitalize their strategies. At the beginning of the study, there were two obvious similarities between the three companies: all three faced serious competitive pressures from Microsoft, and all three faced a challenge in formulating "open" strategies that nonetheless allowed them to retain one or more sources of competitive advantage.

4.1 Responding to the Microsoft Challenge

The reality in the late 1990s was that a single company dominated the IT industry: Microsoft. While it shared the "Wintel" platform control with Intel, unlike Intel Microsoft did not have a competitor in supplying PC manufacturers. Microsoft also played a major role in PC application software, server operating systems, server applications and an increasing role in mobile devices.

IBM, Apple and Sun sought new strategies to respond to pressure from Microsoft. With its PC, IBM had relinquished industry leadership to

Microsoft; Apple had traditionally been Microsoft's primary competitor in 16-bit PC platforms; and Sun found its core workstation and server business threatened by Microsoft's attempts to expand beyond the PC.

Many hardware vendors had benefited from Microsoft's technical leadership. Firms such as NCR, Siemens, and Unisys sold computers built on commodity components, focusing on a particular geographic or market niche. Other companies such as Dell and Fujitsu obtained advantages of distribution or operational efficiency that enabled them to flourish in commodity markets (Kraemer et al., 2000).

But such was not in the "organizational DNA" of these three focal companies. IBM had led the computer industry for 30 years, supporting the huge infrastructure of world's largest computer company. Similarly, as the only surviving PC startup of the 8-bit era, Apple had from its birth pursued a go-it-alone strategy. Finally, Sun, while building upon the success of Unix and the open systems movement, had sought to differentiate itself by having the most complete and fully featured Unix-based operating system.

Until the rise of the IBM PC "clones," IBM had largely controlled its own destiny through proprietary architectures. However, to win allies for their *de facto* standards, IBM — and to a lesser, degree Sun and Apple — had tried to lead formal and *de facto* standards organizations to support their respective technology initiatives, as with IBM's DOS/V PC standard in Japan (West and Dedrick, 2000).

Such alliances were more problematic for open source technologies. In the decentralized "bazaar" model epitomized by Linux, there was no central administrative authority with which to negotiate; to some degree, publishers such as Red Hat and SuSE ended up filling this role.[6] The more centralized cooperative efforts such as FreeBSD or Apache — the "cathedral" in the famous typology of Raymond (1999) — provided a more identifiable authority for negotiating alliances.

4.2 Leveraging Openness While Keeping Differentiation

The open source movement was in part framed as a reaction to Microsoft and its proprietary control of the computer industry, just as the open systems movement a decade earlier had been a reaction to the proprietary control of IBM. As such, the movement found natural affinity with Microsoft's three major platform competitors, as well as hardware makers — particularly Intel — who both benefited from Microsoft's success but also wanted to increase their independence from it. Thus in 2001, HP, IBM, Intel and NEC launched a joint open source research lab in Oregon and Japan; meanwhile, Linux publisher Red Hat had already won equity investments by Intel, followed by Compaq, Dell, HP, IBM and Novell (MacCormack and Herman, 1999; West and Dedrick, 2001).

The problem for IBM, Apple and Sun was that by making source code freely available and modifiable, open source inherently reduced barriers to entry by rivals and switching costs by customers. So despite the appealing logic of mutual adversaries ("the enemy of my enemy is my friend"), a pure open source strategy would eliminate each company's historic source of differentiation, their proprietary software. Each of the firms faced a dilemma of how to adapt an open source strategy suitable for their respective core competencies and resources.

5. Apple: Re-use and Leverage

Apple Computer's heyday had come with its creation of proprietary computer platforms, but — like IBM — its onetime supplier Microsoft had become a formidable rival that threatened this traditional source of advantage. This forced Apple to consider (and embrace) something previously unthinkable: sharing technology through the use of open source.

Apple's adoption of open source came after several failed attempts to develop a new PC operating system, and seemed to

---

6 In analyzing the "bazaar" archetype for the Linux project, one must recognize the ongoing *de facto* leadership and control exerted by founder Linus Torvalds.

offer Apple a way out of its technological dead-end. In 1995, Apple was the first major computer maker to sponsor a Linux implementation for its own hardware. The experimental Linux project was eventually dropped, but it was replaced by a new OS that combined a unique mix of proprietary and open source components.

5.1 Strategic Position in mid-1990s

Although the most successful U.S. maker of 8-bit personal computers, Apple had several false starts before releasing a popular 16-bit PC with the Macintosh. The Macintosh differentiated itself with ease of use provided by a proprietary graphical user interface (GUI), but for various reasons lagged MS-DOS and its Windows in adoption.[7] Finally cutting prices to respond to its MS-DOS rivals, Apple grew market share in major markets in the early 1990s and enjoyed record market share and revenues through its fiscal year ending September 1995. However, the August 1995 release of Windows 95 effectively eliminated its ease of use differentiation. Both the actual and predicted shift in demand helped fuel a downward spiral that in the next two years brought nearly $2 billion in losses and forced resignation of two Apple CEOs.[8]

Ongoing improvements in operational efficiency from 1996-1998, the return of Steve Jobs as CEO in August 1997 and the introduction of the popular iMac one year later at least temporarily quelled predictions of the company's immediate demise. But the company faced the same dilemma as at the beginning of the decade — upgrading its core operating system to incorporate modern multi-processing and memory protection features without rendering its software library obsolete. While Microsoft had released Windows NT as an eventual replacement for Windows 95, Apple was still using improved versions of its 1984 architecture, designed for a machine with 128K RAM and two floppy drives. More seriously, the company's decline in both profits and R&D staff made it even tougher to respond to the Microsoft challenge than it had been in 1990.

As a quick fix, in 1996 Apple evaluated various alternatives to jump-start its future operating system design, including licensing Windows NT and Sun's Solaris variant of Unix. In the end, it bypassed the purchase of the promising but incomplete BeOS to acquire NeXT, a company that Jobs had founded in 1985 after leaving Apple. The acquisition brought not only NeXT's operating system to Apple, but Jobs and his R&D team, which quickly assumed key positions within the company.

5.2 Shifting to Unix

The search for a modern operating system would eventually bring both the Unix operating system and open source to the core of Apple's long term platform strategy. In buying NeXT, Apple cast its future with the NextStep operating system that was the acquiree's primary asset. NextStep was a Unix variant that combined the Mach operating system "kernel" with other components from BSD Unix. To this NeXT had added various extensions, including a graphical user interface, software development and system administration tools.

Prior to the NeXT acquisition, Apple already had experience with Unix, offering the AT&T-based A/UX operating system from 1988 to 1995. But by the time it was abandoned after a shift to IBM's RISC-based processors, A/UX had attracted only 15,000 users — at a time when Apple was selling 4 million units a year (Hess, 1995; Dataquest, 1996).[9]

---

7 The discussion of Apple's proprietary platform strategies is adapted from West (2000).

8 The conventional analysis attributes Apple's problems in the late 1980s and 1990s to its failure to license its operating system to rivals prior to the rise of Windows in 1991. Assessing such *post hoc* advice is complicated by the variety of other strategic and execution errors during this period, including nearly $2 billion spent on unsuccessful efforts both to upgrade its core technology and diversify into new market segments such as handheld computers and set-top boxes (West, 2003).

9 From 1996-7 Apple sold IBM's AIX version of Unix, in between canceling A/UX and buying NeXT.

Apple had experimented with both the Mach OS and open source prior to the NeXT purchase. In 1995, to help its operating systems design efforts it funded a research project to adapt the Mach kernel for Apple hardware. Because of the rising popularity of Linux, Apple selected a Mach/Linux combination instead of the standard Mach/BSD combination. This operating system, named MkLinux, was released in a series of developer releases from May 1996 until the summer of 1998, when the project was destaffed by Apple and handled over to its base of user-programmers. The Mach and Linux source code were released under BSD-style and GNU-style open source licenses.

Instead of continuing with Linux, the 1997 merger with NeXT committed Apple to adapting its existing Mac OS to use the BSD-based NextStep technology. In 1997 it announced "Rhapsody" — a Macintosh version of the existing NeXT OS — which shipped as "Mac OS X Server" in 1999. However, the company's main focus was a comprehensive upgrade of the NeXT technology, incorporating a new GUI and an emulator for older Mac OS programs. This "Mac OS X" was originally promised for late 1999, was released in beta form in September 2000 and eventually shipped to users in March 2001. During 2000 and 2001, the Mac OS X user and Server code bases were merged so that both used the same core technology and user interface.

5.3 Building a New OS on Open Source Parts

The original NextStep was based on both AT&T and BSD licensed Unix code, but by 1999 Apple replaced that code with FreeBSD, one of the three open source Intel-based versions of Unix. More significantly, Apple announced that it was releasing the Mach, FreeBSD and some NeXT components as a new open source operating system, "Darwin," which outside experts described as a new member of the BSD family tree.

Darwin, in effect, was the central core of Apple's Mac OS X Server and subsequent Mac OS X (Figure 1). It provided proven multi-user memory management and process control services which the original Mac OS had lacked. At the time, Apple proclaimed it was "leading the industry by becoming the first major OS provider to make it's [sic] core operating system available to open source developers," (Apple Computer, 1999).

Project manager Ernest Prabhakar noted that a catalyst for the Darwin strategy was pressure from large university customers with specialized networking needs:

> We realized that the pieces they're most interested in are the most commoditized. There wasn't any proprietary technology added that we had to worry about them copying. … We started making the case [that] we should just open the source code and release it as a complete BSD-style operating system (Wayner, 2000: 175).

Prabhakar later said that Apple sought to "embrace and enhance" existing open source technologies, but in some cases would "embrace and layer" by building Apple's proprietary code on top of the publicly shared open source code.

The move was proclaimed by optimists as legitimizing open source development, in that a once proprietary platform firm was willing to share a portion of its core operating system with outside developers. However, other open source advocates attacked some of the exclusions in Apple's license, and the controversy divided previously allied leaders of the open source movement. While the major license concerns were resolved within a month, the dispute hurt open source advocates within Apple by demonstrating the lack of a single spokesperson within the open source community who could speak on behalf of the entire community (Shankland, 1999; Wayner, 2000: 162-163).

Some also complained about Apple's choice of the BSD rather than GPL style license, which allows a commercial firm to take the public source, make proprietary modifications, and to release that part public, part private product as a proprietary solution (e.g., Leibovitch, 2001). Such is both the inherent advantage (for commercial firms) and disadvantage (for open source purists) of the BSD-style license.

But the most enduring controversy continued over Apple's decision to hold some layers of its operating system entirely proprietary. Most of the public Darwin source was a derivative work of the public FreeBSD and Mach, and so as a practical matter Darwin offered the same functionality than its FreeBSD cousin. Apple had held out the largest (roughly 75%) and most valuable parts of Mac OS X — its graphical user interface, the NeXT and Mac OS application support — meaning that Darwin was not a complete GUI operating system and thus of little interest to average users. Other Apple-controlled technologies — such as its TrueType fonts and QuickTime multimedia software — were similarly excluded, preventing Darwin (and Linux) users from using these technologies without Apple hardware.

In March 2002, Apple helped launch a new organization, OpenDarwin.org. OpenDarwin maintained a stand-alone version of Darwin outside Apple's direct control, with its own discussion groups and bug lists, but sharing source code with Apple. By December 2002 it had 48 identifiable user-contributors.

On paper, Apple enjoyed the best of all possible worlds. The open source Darwin allowed it to leverage off the larger BSD communities to incorporate enhancements in networking and other technologies, and to port Unix-based applications such as web and mail servers. Darwin also provided low-level documentation to third party hardware vendors, freeing Apple to concentrate its support efforts on application software. Apple also retained differentiation in the traditional areas where it had mattered most — in graphics and ease of use for its core markets in graphical design and education.

However, by opening only part of its technology — largely corresponding to the existing FreeBSD — Apple made it less valuable to user-contributors. The fewer users that contributed to the Darwin sources, the less benefit Apple realized from its open source strategy.

6. IBM: From Platforms to Applications

Despite its well publicized travails of the 1980s and 1990s, IBM remains the world's largest computer company, as measured both by total sales and employees. It was also the most aggressive of any incumbent computer maker in embracing Linux and open source.

As he announced IBM's 2001 plans to invest $1 billion in Linux, then-CEO Louis Gerstner predicted an end to the era of proprietary platforms that his company had spawned:

> The movement to standards-based computing is so inexorable, I believe Sun—and EMC and Microsoft for that matter—is running the last big proprietary play we'll see in this industry for a good long while (Wilcox, 2000).

While Gerstner might secretly prefer a return to IBM-controlled proprietary industry, the reality was that he found that an open source world — where hardware vendors and customers all had full control over crucial system software — was preferable a proprietary industry controlled by Microsoft.

6.1 Strategic Position in mid-1990s

IBM was best known for creating the first computer platform with its S/360 proprietary mainframes, and its subsequent AS/400 minicomputer platform. Many argued, however, that its most successful architecture was the one that got away: the IBM PC. The shift of leadership was traumatic for IBM. In the early 1990s, IBM spent billions of dollars on OS/2 and joint ventures with various industry rivals — all in an unsuccessful attempt to re-assert its proprietary leadership of the computer industry, or at least break free of dependence on Microsoft. By 1996, the hatred of the "Evil Empire" within IBM bordered on the profane (Garr, 1999: 187-188).

After losing the PC OS war, IBM's leadership was confined to those segments (large computer systems) that were enduring a systemic decline. In more rapidly growing

PC and workstation markets, it was an also-ran. In response, the company sought to reposition the demand for mainframe computers, first by promoting client/server architecture and later by developing software to enable them to serve as massive web servers. Meanwhile, it placed increasing emphasis on the sale of software and services, winning business based on its unmatched ability to offer a complete end-to-end "turnkey" solution.

6.2 Phase I: Applications

IBM's first major open source initiative came in its efforts to integrate corporate mainframes (with their vast legacy databases) to directly support e-commerce and intranet initiatives. In June 1998, it unveiled its WebSphere product family, which built upon the Apache open source web page server. IBM had begun with its own internally-developed web server, but adopted the Internet's most popular web server after failure of negotiations with Netscape over licensing its proprietary web server (McKay, 1998).

IBM's efforts to adapt Apache to meet its specific needs set a pattern for collaboration in its sponsorship of subsequent open source efforts. IBM helped fund Apache's adaptation for use on Windows NT, because it was central to its WebSphere strategy (Moltzen and Burke, 1998). Meanwhile, IBM engineers contributed code back for use by all Apache implementations, and IBM hired one of the key open source developers to act as a permanent liaison (Wayner, 2000: 181-183). IBM found that working with (largely user-driven) open source group provided more flexibility than using someone else's proprietary solution, a lesson it would later apply to Linux.

WebSphere was also the first IBM application to be made available on various Linux platforms. IBM added a Linux version of its DB2 database, which in December 1998 was released in beta form on IBM's web site; IBM later ported a third major product, its Lotus Domino groupware. As with other Unix application vendors, IBM found that offering a Linux version required a comparatively small investment.

The WebSphere product indirectly led to IBM's November 2001 formation of Eclipse, an independent open source consortium to develop common software tools. IBM donated source code it valued at $40 million to launch the consortium, which eventually grew to include Fujitsu, HP, Oracle, Red Hat and SAP. Sun (with its competing JavaBeans) was notably absent.

While IBM's initial focus was on developing web-enabled applications in Java and HTML, the Eclipse project bragged that it was a "universal tool platform … for anything and nothing in particular" (Eclipse.org 2002). The IBM-developed architecture allowed third party developers to write plug-in modules to support various programming languages (C, C++, Cobol), file formats and external software products (like IBM's WebSphere Application Server).

The code was licensed under the Common Public License, an open source license which like BSD — and unlike GPL — allowed commercial distribution of derivative works. As Eclipse evolved, IBM merged updates from it into its commercial product, WebSphere Studio Workbench. In December 2002, IBM announced a $2.1 billion purchase of Rational Software, an Eclipse consortium member that sold application development tools.

Overall, IBM's development of applications using open source software had three common threads. First, IBM accepted commodization of certain layers of its application architecture and was thus was willing to collaborate with open source software programmers to make a shared technology available to all; these layers typically implemented open Internet standards which offered less opportunities for differentiation. Second, in many cases the shared software competed with proprietary solutions developed by Microsoft using its $4 billion annual R&D budget, such as its Internet Information Services web server. Finally, the shared software was released under a non-GPL license allowing IBM to retain technology or make proprietary enhancements.

6.3 Phase II: System Software

Unlike with applications, IBM's direct support for Linux as a replacement operating

system was long in coming. Throughout 1998, IBM's corporate strategists had no intention of providing or supporting Linux, as it would reduce differentiation and also threaten its high-margin proprietary operating system sales. So instead of being officially sponsored by developers of IBM's S/390 mainframe, Linux was successfully "ported" in late 1998 through unsanctioned effort of programmers at IBM Germany (Hall, 2001). This internal version of S/390 Linux was eventually released for customer downloads in February 2000. Subsequently, the changes to standard Linux to support the S/390 were made available both on IBM's web site and through commercial distributors Red Hat, SuSE and TurboLinux. IBM also endorsed existing Linux distributions for its PC servers.

In early 2000, IBM announced it would support Linux across its entire range of servers, from PCs up to the largest mainframe (Table 5). At the same time, a reorganization eliminated its Internet business unit, and reassigned its head, Irving Wladawsky-Berger, to head a combined Unix-Linux unit. That group was located within IBM's Enterprise Systems division — clearly targeting IBM's traditional large corporate customers rather than the PC-based Internet service providers that up to that point had been the largest market for Linux.

Beyond the level of customer demand (or "market pull"), endorsing Linux gave IBM key strategic advantages. First, Linux provided a common set of APIs across its entire product line, providing a unified architecture for software developers. Second, the comparatively immature (yet complex) operating system required support services, a traditional IBM strength, as Wladawsky-Berger later explained:

> We've wedded ourselves to the integration of the solution, the notion being that the Internet and e-business solutions are more important than any particular component. And as a result, we've changed all our business models so that the integration of the pieces has become more important than any one piece (Cooper, 2001).

Finally, the open source operating system allowed IBM to make changes to improve its hardware differentiation for enterprise customers. As chief technology officer for PC servers noted, Linux "has given IBM an opportunity we didn't have before to play to our strengths, which is availability and reliability" (Shankland 2002).

Wladawsky-Berger also said that the open source model was a logical extension of the long-standing IBM research culture. Towards that end, in August 2000 IBM's special developerWorks open source web site released the source code from two large IBM research projects, the Andrew file system (later called OpenAFS) and its Jibes Java compiler. Both were released under the IBM Public License, an antecedent of the Common Public License.

7. Sun: Opening New Platforms

Sun was founded in 1982 to make engineering workstations, and by the late 1980s had outlasted its rival Apollo to lead the market. While its Sun OS (later Solaris) platform was based on the same UNIX operating system adopted by most of its workstation rivals, Sun successfully differentiated itself from other rivals through ongoing enhancements in its operating system, particularly with its support for data networking (Garud and Kumaraswamy, 1993).

As the 1990s ended, Sun faced threats to its core business, by open source Linux on the one hand and Microsoft's Windows NT on the other. At the same time, it sought to retain its traditional control of the operating system and other technology that had fueled its success in the workstation and server market.

7.1 Strategic Position in mid-1990s

In the mid-1990s, Sun held a strong position in workstations. Meanwhile, it was well positioned to capitalize on rapidly growing industry demand for midrange servers and those that required networking and Internet support. While Sun's marketing had historically emphasized support for "open architecture," it used proprietary extensions to Unix software to differentiate

itself from workstation rivals such as HP, IBM and DEC (later Compaq).[10]

To improve adoption, it licensed its workstation and OS technology to customers and complementors; this included a small number of makers of "clone" products, most notably in Japan (Garud and Kumaraswamy, 1993). At the same time, Sun retained full control of the architecture, allowing it to rapidly evolve the technology rather than negotiate with standards committees. As such, Sun's strategy more closely fit the "proprietary but open" model of Morris and Ferguson (1993) than did Microsoft.

By concentrating on Unix-based systems and ignoring PCs, in the mid-1990s Sun held a unique position in the computer industry with respect to Microsoft. Most computer companies licensed one of Microsoft's Windows operating systems for servers, PCs, laptops or handheld computers; while Apple notably did not, it bundled Microsoft's web browser and actively courted its Office application suite. As such, no matter how much some initiatives of IBM or Apple (or HP or Compaq) might conflict with Microsoft, at other times they were Microsoft allies. Even bitter rivals America Online and Netscape made Windows support their top applications priority.

By contrast, Sun lacked such a "co-opetition" relationship with Microsoft. A decade earlier, the two firms had little overlap with completely different technologies and customers. However, industry trends — particularly the shift of the Internet from a research network to a consumer one — had led them to become direct competitors (Goff, 1999). The Windows NT server operating system was directly aimed at the Unix server business led by Sun, while any success of Microsoft in establishing proprietary Internet protocols would come at Sun's expense. Meanwhile, Sun did not ship Microsoft's operating system or applications. Thus, it was not surprising that Sun's co-founder and long-time CEO Scott McNealy was one of Microsoft's harshest and most vocal critics, both because of the two company's conflicting goals and because of the lack of dependence on Microsoft for any key technology.

Both Windows NT (later Windows 2000 and XP) and Linux posed a low-cost threat to Sun's lucrative server business: both were based on high volume Intel processors, whose performance was increasing more rapidly than that of Sun's proprietary RISC processors. In another dimension, Windows and Linux were attacking Sun from opposite sides: the former represented a more proprietary approach under control of a strong, centralized rival, while the latter offered greater openness that was supported (initially) by a diffuse group of hobbyists.

7.2 Strategy 1: New Platforms

Sun's primary strategy during the late 1990s was to establish new platforms independent of Microsoft that would limit (if not reduce) Microsoft's control of industry standards.

Most of Sun's efforts went towards establishing Java as a new platform with a common set of APIs available on a wide range of computer systems, under the slogan "write once, run anywhere." An early prototype of Java was made available for user downloads in May 1995, and licensed by most major computer companies over the next year (Garud, Jain, Kumaraswamy, 2002). Sun mounted a four year lawsuit accusing Microsoft of trying to hinder Java's success, a lawsuit settled out of court in January 2001.

After distributing previews of its technology and generating great interest, Sun spent three years (1997-2000) trying to get Java established as a *de jure* standard albeit under Sun's control, first at the ISO/IEC Joint Technical Committee 1 (JTC1) and then later at ECMA.[11] Both efforts were withdrawn after vigorous and well-financed opposition from competing computer makers, particularly Microsoft and HP (Egyedi, 2001). Sun also objected to the

[10] However, the differences in Unix APIs increased costs for user and software developers, eventually forcing vendors to agree on a common GUI and system specifications in the mid-1990s, ending the "Unix wars".

[11] ECMA was founded in 1961 as the European Computer Manufacturers Association, but in 1994 switched to its acronym in hopes of increasing its global influence.

provisions that would have required it to surrender IPR to the standardization committee.

In a second major initiative, in August 1999 Sun spent $73.5 million to acquire the German maker of StarOffice, a clone of Microsoft Office. As Sun later noted "It is critical for all of Sun's customers that there be open, viable, cutting-edge office productivity software available to run in the heterogeneous network and across all platforms" (Sun Microsystems, 2001). The company also stated a desire to shift the industry from the purchase of software products to the rental of network-intensive application services (which would require more Sun systems to implement).

To a large degree, Sun's preoccupation with its offensive strategy against Microsoft contributed to its failure to defend against Linux. For example, in a 3,000+ word 1996 interview, Sun's CEO did not mention Linux or open source code at all. Instead, he reiterated Sun's historic push for Unix-based open systems over proprietary platforms:

> Q: Now Microsoft's recent licensing of Java seems like a solid vote of confidence in the technology, but are you worried Microsoft might try to position Java as just another language, or bury it under ActiveX?
>
> McNealy: We're always worried people will try and hijack the standards on the network… The real beauty of the Net is all interfaces are open, they're multivendor and you can publish your data or publish your application once and know it will run on everything. That makes it a lot more competitive and lowers the price to the user. That's not exactly what all the technology companies want to do. They want to get you locked in (Taylor, 1996).

Ironically, the efforts of Microsoft and other proprietary computer makers to derail Sun's platform strategies led it towards its first open source strategies, which improved its ability both to compete and cooperate with open source software.

7.3 Strategy 2: Partly-Open Source

Instead of responding to Linux, Sun's first open source strategies focused on its competition with Microsoft in getting core technologies adopted by users and software developers. The evolution of these strategies began with Java, extended to StarOffice and eventually reached its core Solaris operating system. In addition to its pro-active strategies, Sun also reacted to pressure to increase the access to its Java code exerted by licensees and standardization committees.

The effort to define Sun's open source strategy was led by chief scientist Bill Joy, who had been one of the leading engineers in the Berkeley Unix group before leaving to co-found Sun (Kim, 1999). The most radical option — which would please the largest number of open source advocates — would have been to use the GNU Public License, in which all changes made by Sun, its customers or competitors must be shared with everyone. While this would fuel adoption, it posed real concerns about appropriating the returns of Sun's R&D investment, as Joy explained:

> I can't license all of Sun's intellectual property under the GPL, because it just won't work. I don't see any reason why I should give somebody who's doing commercial reuse unfettered access to stuff that cost me millions of dollars to do. We're spending over a billion dollars a year in research. I can't just throw it all on the street.…
>
> If I make code available under the GPL, I'll lose control of it. … The GPL just doesn't solve my business problem at Sun. I would like all of our intellectual property to be available in source form, but I can't economically do that under the GPL (Kim, 1999).

Instead, in February 1999 Sun released Java source code under what it called the Sun Community Source License, a hybrid between a traditional proprietary license and a BSD-style open source license. The license had four basic elements: 1) right to modify

the source code; 2) royalty free distribution in open source projects; 3) royalties for commercial redistribution; 4) testing requirement to maintain compatibility and prevent forking (Loukides, 1999). From Sun's standpoint, the license: "provides protection for intellectual property, … guarantees structured innovation within a single responsible organization, [and provides] clear control over compatibility" (Gabriel and Joy, 1998). The Sun license is innovative in its governance: the SCSL processes in many ways resemble a formal standards consortium or *de jure* standardization committee more than the "bazaar" more associated with decentralized Linux development.

If Sun had concern over rivals and control with Java, these were even greater with Solaris, its Unix-based operating system that had provided its key differentiation in the workstation and server market. However, in October 1999 it announced that it would release Solaris under the restrictions of the SCSL, and in December 2000 finally did so.

Sun adopted a different strategy for StarOffice; in October 2000 it released all source code to a new organization, OpenOffice.org. The code was licensed in a way that guaranteed the source would always remain public, but allowed its use in commercial products by Sun or anyone else.[12] After nearly a decade of development, StarOffice badly lagged Microsoft's product in features, compatibility and reliability. The "bazaar" community of open source user-developers was ideally suited for addressing such concerns, as Sun explained:

> By engaging the energy and creativity of developers worldwide, we will accelerate the addition of innovative features and improved integration with other products. Making the source code available also enables the StarOffice software functionality to be ported to a wider range of systems (Sun Microsystems, 2001).

In 2002, Sun released a $76 commercial version of StarOffice, sharing code with OpenOffice, which remained an ongoing open source project. Both Sun and the open source community retained a common goal, being able to access business documents compatible with Microsoft Office and the dominant Windows standard.

7.4 Strategy 3: If You Can't Beat Them, Join Them

Compared to its other workstation rivals — particularly HP and IBM — Sun did little to embrace Linux. From 1998-2000 it provided technical assistance to outside groups porting Linux to run on its 32-bit SPARC and 64-bit UltraSparc line of Solaris-based systems. It did not sell systems with Linux pre-installed, leaving that to value-added resellers. In fact the demand for Linux on Sun hardware was so weak that in November 2000 a leading Linux distributor, Red Hat, canceled future development of Linux for Sun computers.[13]

Sun was initially ambivalent about the success of Linux. On the one hand, increasing adoption of Unix-based systems reduced Microsoft's influence and improved availability of Unix-related software, training and engineers. On the other hand, Sun had charged a premium to its workstation rivals based on its superior Solaris software, so a world where Linux was the norm would eliminate that advantage (Rosenberg, 2000). As one analyst put it: "Sun is giving Linux some rhetoric, but Sun does not want Linux to take off" (Scannell and Gardner, 1999).

Eventually, in the face of competition in the low-end server business, Sun adopted Linux as a server operating system, albeit to a lesser degree than IBM. In 2000, Sun purchased Cobalt, a maker of low-end Linux server appliances. Two years later, Sun

[12] Technically, the Lesser GNU Public License has some characteristics in common with the GPL, but in practical terms it was more similar to an Apache-style license in that it lacked the "viral" provisions of the GPL.

[13] It should be noted, however, that Sun's Internet servers incorporated the same Open Source applications (such as Apache and sendmail) as the other Unix and Linux-based servers.

announced plans for its own branded Intel-based servers, running Linux or Solaris.

8. Effect of Open Source on Platform Strategies

The popularity of open source operating systems created both problems and opportunities for proprietary platform vendors. Disclosing software technology through open source licenses would naturally lead to the commodization of such software. Not surprisingly, the response of leading industry firms varied depending on whether they had used software as a source of competitive advantage, and whether they retained other sources of competitive advantage.

8.1 Comparing Strategies by Apple, IBM and Sun

While IBM was one of the world's top software vendors, its software was normally sold as part of a combined solution with its own hardware. It continued to differentiate itself based on services such as integration, services that would provide switching costs if it adopted commodity software. Also, its key revenues were in mainframe and midrange systems where there were few remaining competitors. Under such conditions — with either service differentiation or few viable competitors — software commoditization would be less of a concern, and anything that reduces costs or increases demand is an unvarnished plus. In fact, Linux offered IBM something it had never had — a common set of software APIs across its entire product line.

Linux successes would also hurt Apple and Sun, which historically bundled average hardware with better than average operating systems. Apple and Sun faced diminished profit margins if they shared the same software as its rivals (or vice versa). But unlike IBM, both Apple and Sun released source code from their primary operating system. The two firms adopted two different approaches: opening parts vs. partly open. In part by building on code that was already open source, Apple chose to grant all rights to a subset of its new OS X operating system. Meanwhile, Sun released the entire source of Java and Solaris under restrictive terms — the former to improve adoption, the latter in response to competition from the open source Linux.

All three firms sought to maintain control of their proprietary OS and other technologies, in part to assure that they would continue to evolve and remain competitive. They also had specific concerns about aiding rivals and an historic aversion to sharing profits with others in their value chain.

Prior to embracing open source strategies, all three companies had extended the value of their platforms through proprietary applications and "middleware". Such software had enabled them to serve their respective markets — large corporate servers for IBM and Sun and multimedia-savvy consumers for Apple. By retaining this software as proprietary and unique to their respective platforms, the firms were able to retain at least some differentiation relative to both proprietary and open source competitors.

8.2 Microsoft's Response

Of the industry's largest firms, Microsoft clearly had the most to lose by the having free software supplant commercial operating systems and application software; it lacked the hardware and services revenues to replace software sales lost to free software. On the other hand, Microsoft's proprietary platform strategies continued to be successful and thus it faced the least pressure to adopt an open strategy. However, the rising publicity associated with open source and the potential shift of server customers from Windows to Linux forced it to respond.

In the first half of 2001 key Microsoft executives publicly attacked the movement, particularly the "viral" nature of the GPL (West and Dedrick, 2001); the company later clarified its position to emphasize support for BSD-style licenses. Microsoft also unveiled its own form of licensed source code disclosure called "Shared Source". By 2002, its strategy had evolved to allow PC vendors, third-party developers and large end-users to view but not modify the source to Windows. To win the hearts and minds of academics, Microsoft also

allowed universities to both view and modify the source for internal research.

The role of Microsoft was clearly paramount in the open source license strategies adopted by both nonprofit and corporate software developers. Linux developers and other backers of the GNU license often cited Microsoft's decision to use BSD networking technologies in Windows, then enhance those technologies in a way that made Windows incompatible with Unix-like systems. As Sun's Bill Joy complained:

> The top predator now is Microsoft. We didn't have a top predator back when I did TCP/IP. When you have a person with unlimited funds who is clearly focused on destroying the value proposition of what you're doing, you'd be a fool not to account for them in the strategy that you adopted (Kim, 1999).

9. Discussion

The use of multiple qualitative case studies provides a rich opportunity for building theory in emergent areas that is grounded in empirical data. This section uses induction to generalize the observed open source platform strategies of proprietary firms into broader theoretical predictions about competitive strategies for IT platforms, and suggest areas for future research.

Such theory always runs the risk of being idiosyncratic and not generalizable to the entire population (Eisenhardt, 1989). In this case, by studying firms that were previously successful with proprietary strategies, such findings may not be applicable to firms that unsuccessfully pursued proprietary strategies, or *de novo* entrants that lack prior platform capabilities upon which they can build. There is also the risk of attempting to generalize from a still-emergent process: the adoption of open source — both by business end-users and proprietary hardware companies — is still comparatively recent phenomenon. Any or all of the companies studied could fail in their efforts, or find greater success by returning to more proprietary strategies.

9.1 Shifting from Proprietary to Open Source Strategies

The study suggests a three stage evolution of proprietary platform vendors to the use of open source.

Proprietary Platforms. As with other industries, computer industry pioneers began by vertically integrating to deliver a complete proprietary platform solution. Whenever possible, the firms prefer proprietary platform strategies, because they provide better barriers to imitation and better margins. But as noted earlier, this strategy may only be available to one or two market leaders.

Open Standards. For many IT vendors, the use of proprietary platform strategy becomes infeasible for some combination of technical and economic reasons, and they modify their platforms to incorporate open standards that are shared with one or more competitors. Among the motivations:[14]

- market share lower than the minimum efficient scale necessary to support proprietary R&D;
- not enough market power to resist buyer demands for open standards;
- "tipping" of the standards contest in favor of the open standard, making it infeasible to establish (or maintain) a proprietary standard; or
- a decision to accept commodization of the particular architectural layer and shift competitive advantage to another architectural layer.

These criteria are consistent with von Burg's (2001) analysis of the adoption of Ethernet as the preferred networking technology by both proprietary and open systems vendors.

While shifting to such shared platforms may be the most cost-effective solution, it can be difficult when it runs contrary to the corporate culture and previously valued core competencies. The experience of Apple, IBM and Sun suggests that shifting to even a

[14] In the case of the PC industry, the shared multivendor architectural layers are proprietary technologies provided by Microsoft and Intel, which addresses some but not all of the pressures for open multivendor standards.

partly-open architecture may require a major external shock to force firms to relinquish previous innovation-driven differentiation strategies.

The shift also contains unexpected pitfalls. Firms with a successful proprietary architecture are able to simplify their technical and business decisions, because they control their environment and don't have to interoperate with the rest of the world. When the proprietary strategy fails, firms are forced to work with open standards to achieve interoperability, and such interoperability both requires additional technical efforts and also reduces the lock-in of existing customers. For example, faced with plummeting market share, Apple abandoned its proprietary peripheral interface standards and switched to industrywide standards, discontinuing most of the proprietary peripherals that once accounted for much of its revenues (West, 2000).

Having lost various platform battles with Microsoft, both Apple and IBM have been forced more than ever before to shift from proprietary to open standards. Sun's business model had always required co-existing with open standards, but now has relinquished control of more platform standards to "compete on implementations."

Without innovation and proprietary lock-in to provide barriers to entry and imitation, invariably firms will find it difficult to achieve competitive advantage with these new strategies. Among the functional strategies that the three firms used include marketing, customer service, product design, engineering efficiency and leveraging previously establish brand name reputations; the long-term viability of all these strategies have yet to be proven.

Open Sources. The transition to an open source platform strategy is a continuation of that to open systems, driven by the many of the same factors. Open source, however, eliminates the ability of vendors to compete based on implementations since the details of an implementation are visible to all.

A vendor's decision to disclose technology is an irrevocable waiver of its ability to appropriate the returns from that technology. The use of hybrid strategies suggests that proprietary vendors are aware of the competitive risks of such an appropriability waiver and are thus experimenting to find the right compromise between totally proprietary platforms (which would be rejected by the market) and totally open ones (which would eliminate all competitive advantage). Thus far, the two hybrid strategies have been (Figure 2):

- *opening parts*, waving control of commodity layer(s) of the platform, while retaining full control of other layers that presumably provide greater opportunities for differentiation;
- *partly open*, disclosing technology under such restrictions that it provides value to customers while making it difficult for it to be directly employed by competitors.

The former strategy is important as an offensive strategy to speed adoption of a new platform-related standard or a particular implementation of such a standard. Waiving intellectual property rights makes the standard (or implementation) more attractive to competitors and key users, priming the positive-feedback bandwagon effects that can accrue to early market leaders. It also increases the number of products that are interoperable with the vendor's products, particularly important for networking and other communications standards.

Both approaches allow sophisticated users (such as large business enterprises, universities or IT industry suppliers) to help improve the products they use, consistent with the long-identified role of technically knowledgeable industrial users (von Hippel, 1976).

9.2 Future Platform Strategies

The study of standards and standards architecture competition has focused on three basic approaches:

- vertically integrated proprietary systems, as represented by the IBM 360 (Chandler, 1997; Moschella, 1997);
- platforms assembled from proprietary layers that are freely licensed to all, such as "Wintel" architecture (Morris and Ferguson, 1993; Grove, 1996); and

- *de jure* standards not sponsored by any single firm but shared by all, epitomized by "open systems" and Europe's GSM digital telephone standard (Gabel, 1987; Funk and Methé, 2001).[15]

Open source standards differ from other unsponsored open standards mainly in degree, to the extent that the entry and imitation barriers are dramatically lower. But the idea of a shared standard — with the associated implications for governance and differentiation — is not fundamentally different between the open source Linux or FreeBSD and its open systems (Unix clone) ancestors.

To a lesser degree, hybrid platform strategies have existed for decades, driven by the ever-increasing need for systems interoperability between or within organizations. Even the most proprietary platform incorporates open industrywide standards such as ISO character sets, ANSI C, Ethernet or TCP/IP. Other firms (notably IBM and Microsoft) have taken portions of their platforms and gotten them adopted as industrywide standards.

The hybrid strategies of Sun and Apple blur the lines between the proprietary and unsponsored standards. By retaining an element of control, they retain many of the competitive benefits of sponsorship. However, by reducing duplicative R&D they can create shared communities that in many ways are indistinguishable in practice from nominally unsponsored standards — assuming that the sponsors move aggressively enough to build sizable communities of adopter/collaborators.

The open source strategies studied also call attention to the use of platform extension as a strategy to deal with commoditization of lower-level platform layers. For computer platforms, such extension normally involves developing additional application or "middleware" as the highest level standards layers of an architecture. While Apple had from the beginning differentiated its operating system through its ease of use, after that advantage disappeared in 2001, Apple began bundling consumer applications to differentiate its platform among PC buyers. Microsoft began such bundling even earlier, when it included its Internet Explorer application with its Windows 95 operating system.

Such vertical integration into applications suggests at least a partial re-examination of the assumption that platforms succeed through their ability to attract a supply of third-party applications. Gallagher and Park (2002) have shown that in-house applications development was crucial in deciding a success of platform contests in the videogame console industry. If this pattern is more generally applicable, then it suggests firms need to garner the financial resources to supply a complete (or at least basic) supply of complementary assets expected by adopters, rather than building early market share perceptions to attract third party suppliers of such assets.

In other cases, the attempt to differentiate may continue not to higher architectural layers, but with system integration or design. In November 2001, Nokia announced a plan to license its PDA/mobile handset applications to rivals, either for use with their own OS, or to run on a multivendor OS developed by the Symbian joint venture (Nokia, 2001).

9.3 Implications for Open Source Development

How open is open enough? Open source provides few direct benefits to the vast number of users who lack the requisite technical skills to do their own development, but instead is best suited for technically proficient users (such as Internet service providers) with strong motivations for customization (West and Dedrick, 2001).[16]

---

15 Note, however, that the nominally open GSM standardization effort led by Nokia and Ericsson built upon patent portfolios that were used to exclude Japanese vendors from the European market (Bekkers et al., 2002).

16 Direct benefit from access to open source code would appear to require some combination of programming skill, (personal or professional) motivation and (personal or organizational) slack time. In their study of Apache users, Franke and von Hippel (2002) found highly heterogeneous requirements among users (motivating customization), and also that professional skill predicted customization activities.

The degree to which open source adds value beyond this niche depends on how much it enables other attributes more directly valued by users, such as greater reliability, lower cost or expanded variety of complementary assets.

An example of such indirect benefits is the provision of complementary assets. Normally, the provision of applications for an operating system is controlled by the formal, published interfaces (Langlois and Robertson, 1995; West and Dedrick, 2000). However, in an open source system, a third party software supplier can add its own interfaces as needed to provide functionality unanticipated by the original author of the OS. Apple specifically opened the lowest layers of its Mac OS X architecture to enable user support of unusual third-party hardware configurations.

Another postulated indirect user benefit of open source systems is increased reliability through the concurrent debugging efforts of a widely distributed community of user-programmers (Raymond, 1999). Such activity appeared to be an important goal for all three firms studied, as the proprietary vendors sought volunteer labor to find and correct some of the gaps in their systems. Kogut and Metiu (2001) report that more than 70% of the Linux and Apache contributors had made only one change to the source code tree. This pattern is consistent with users getting involved only when there is a problem of great concern to them.

Research on large, successful open source projects such as Linux and Apache assumes away variance in what may be a key independent variable: size of the user-programmer community. Anecdotal evidence suggests that the major reason open source projects fail is a lack of user-contributors to do the work. In this case, sponsors of open source projects (whether organizations or individuals) vendors face a particularly important adoption challenge: to attract enough of the right sort of users early enough to improve the quality and features of the software.[17]

How applicable are these benefits to the hybrid strategies? For the "opening parts" strategy, must the open part of the platform have value on its own to win enough user-programmers? Or is it enough that it be part of a larger system of crucial importance, as with Apple's release of the Darwin portion of its mainstream Mac OS X operating system?

A "partly open" strategy — such as Sun's Community Source License or Microsoft's Shared Source — must also attract attracting user-programmers. In this case, the question is whether the strategy provides a stable enough allocation of the returns of innovation between software vendors and users. Such strategies presume that there is an intermediate level of disclosure and granting of rights (between fully open and fully proprietary extremes) that will be valuable to users without compromising the IPR owners' competitive concerns. One such intermediate point — the GNU Public License which grants nearly all rights except the use in competing proprietary products — has high value to users because of the complete and irrevocable disclosure.[18] But the other forms used thus far seem to be discounted by most users (particularly those with more fully open options) and thus provide little if any adoption incentive.

---

17 In the adoption typology of Rogers (1995), sponsors would want to attract innovators as the user-programmers, early adopters as non-programming beta testers who would be tolerant of bugs and missing features, so that a complete solution could be developed suitable for the early majority.

18 I recognize that advocates of the GNU-style license would argue that their license is more "free" than the BSD alternative, it that guarantees that all derivatives will also be free. However, such perpetual freedom is enforced by restricting the rights granted with the software, while the BSD-style licensing has few if any restrictions. An economic analog is the choice between giving a child a trust fund or cash; the former assures that the original goal will always be met, while the latter gives the recipient full right to do as (s)he sees fit.

9.4 Future Research

The hybrid open source strategies of proprietary platform vendors suggest that additional research is necessary on a crucial trade-off in technological innovation: resolving the conflicting imperatives of making an innovation successful and profiting from that success.

In his analysis of this trade-off, Teece (1986) focuses on the case where firms face weak appropriability regimes and thus must share the returns of their innovation. Subsequent analyses largely assume that strong appropriability is good, to be sought, and that lack of appropriability is a less desirable condition. Strategists consider how firms should share the economic surplus they have generated, but not the possibility that firms should irrevocably abandon some or all of that surplus by waiving all future appropriability from a given innovation.

However, proprietary software enjoys one of the tightest appropriability regimes available, at least in countries that enforce IPR laws. Meanwhile, the three decades since the invention of the microprocessor offer numerous examples where innovators enjoyed tight appropriability, but the lack of adoption of their innovation meant that there were no returns to appropriate.

The conversion of proprietary software to open source by IBM, Apple and Sun — forfeiting a portion of their IPR — suggests that the more general question of voluntarily waiving appropriability bears further study. Among the issues that might be considered:

- Is forfeiting IPR more or less desirable in cases where competitors (or substitutes) have weak appropriability?
- Is such a differentiation strategy more effective when competing with an organization with tight IPR control? Or does it have general applicability as a strategy to pre-empt adoption of competing technologies?
- Is it the effectiveness dependent on the nature of the technology, its use, or the importance of complementary assets? Are there any other adoption process characteristics relevant to deciding on such strategies?

Other possible areas for future study of open standards include:

- Are there any conditions which are sufficient for predicting whether a vendor should switch from proprietary to open standards? In particular, will the pressure for openness remain confined to less successful vendors, or will it eventually reach all firms?
- Are there categories of standards (e.g., communications infrastructure) that will always be open? Are there any (e.g., document internal file formats) that will always be closed?

Research on open source strategies by commercial firms might include:

- Under what conditions will open standards lead to the widespread adoption of open source implementations by commercial vendors?
- How durable are various approaches (including opening parts, partly open) for creating barriers to imitation using open source software? Are some less likely to create competitive advantage because they don't add perceived customer value, or conversely provide too much aid to competitors?
- Is the decision between open source and proprietary strategies a matter of managerial discretion? Or is there a normative "best practice" for similar situated firms that dictates the appropriate choice for each context?
- When firms select a "partly open" strategy, what is the appropriate governance for vendor sponsored open source communities? Can the conflicting goals of the two parties be reconciled?

References

Anchordoguy, Marie, 1989, *Computers Inc.* (Harvard University Press, Cambridge, Mass.)

Apple Computer, 1999, Mac OS X Server Embraces Open Source With Launch of Darwin, Press release, 16 March, www.apple.com (accessed 27 Dec. 2001).

Apple Computer, 2000, *Mac OS X: Kernel Environment* (Apple Computer, Cupertino, Calif.).

Arthur, W. Brian, 1996, Increasing Returns and the New World of Business, *Harvard Business Review* 74 (4), 100-109.

Beggs, Alan, Klemperer, Paul, 1992, Multi-Period Competition with Switching Costs, *Econometrica* 60 (3), 651-666.

Bekkers, Rudi, Duysters, Geert, Verspagen, Bart, 2002, Intellectual property rights, strategic technology agreements and market structure. The case of GSM, *Research Policy* 31 (7), 1141-1161.

Besen, Stanley M., Farrell, Joseph, 1994, Choosing How to Compete: Strategies and Tactics in Standardization, *Journal of Economic Perspectives* 8 (2), 117-131.

Bresnahan, Timothy F., Greenstein, Shane, 1999, Technological competition and the structure of the computer industry, *Journal of Industrial Economics* 47 (1), 1-40.

Chandler, Alfred, 1997, The computer industry: the first half century, in: David B. Yoffie (Editor), *Competing in an Age of Digital Convergence* (Harvard Business School Press, Boston) pp. 37-122.

Chposky, James, Leonsis, Ted, 1988, *Blue Magic: The People, Power and Politics Behind the IBM Personal Computer* (Facts on File, New York).

Cooper, Charles, 2001, IBM's big thinker, CNET News.com news.cnet.com 22 August (accessed 31 Dec. 2001).

Dataquest, 1996, Worldwide PC Market Grew 24 Percent in 1995, Press release, www.gartnerweb.com, 29 January (accessed 4 Jan. 1997).

DiBona, Chris, Ockman, Sam, Stone, Mark (Editors), 1999, *Open Sources: Voices from the Open Source Revolution* (O'Reilly, Sebastopol, Calif.).

Eclipse.org, 2002, Eclipse Project FAQ, 19 November , www.eclipse.org (accessed 18 Dec. 2002).

Egyedi, Tineke M., 2001, Why Java™ was -not- standardized twice, *Computer Standards and Interfaces* 23 (4), 253-265.

Eisenhardt, Kathleen M., 1989, Building Theories from Case Study Research, *Academy of Management Review* 14 (4), 532-550.

Ferguson, Charles H., Morris, Charles R., 1993, *Computer wars: how the West can win in a post-IBM world*, Times Books, New York.

Flamm, Kenneth, 1988, *Creating the computer: government, industry, and high technology* (Brookings Institution, Washington, D.C.).

Franke, Nikolaus, von Hippel, Eric, 2002, Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software, Working Paper #4341-02 (MIT Sloan School of Management, Cambridge, Mass.)

Funk, Jeffrey L., Methé, David T., 2001, Market- and committee-based mechanisms in the creation and diffusion of global industry standards: The case of mobile communication, *Research Policy* 30 (4), 589-610.

Gabel, H. Landis, 1987, Open Standards in Computers: The Case of X/OPEN, in: H. Landis Gabel (Editor), *Product Standardization and Competitive Strategy* (North-Holland, Amsterdam).

Gabriel, Richard P., Joy, William N., 1998, Sun Community Source License Principles, www.sun.com (accessed 30 Dec. 2001).

Gallagher, Scott, Park, Seung Ho, 2002, Innovation and Competition in Standard-based Industries: A Historical Analysis of the U.S. Home Video Game Market, *IEEE Transactions on Engineering Management* 49 (1), 67-82.

Garr, Doug, 1999, *IBM redux: Lou Gerstner and the business turnaround of the decade*, HarperBusiness, New York.

Garud, Raghu, Jain, Sanjay, Kumaraswamy, Arun, 2002, Institutional Entrepreneurship In The Sponsorship Of Common Technological Standards: The Case Of Sun Microsystems And Java, *Academy of Management Journal* 45 (1), 196-214.

Garud, Raghu, Kumaraswamy, Arun, 1993, Changing competitive dynamics in network industries: An exploration of Sun Microsystems' open systems strategy, *Strategic Management Journal* 14 (5), 351-369.

Goff, Leslie, 1999, Sun and Microsoft go public, *Computerworld*, 20 Sept., p. 94.

Greenstein, Shane M, 1997, Lock-in and the Costs of Switching Mainframe Computer Vendors: What Do Buyers See? *Industrial and Corporate Change* 6 (2), 247-274.

Grove, Andrew S., 1996, *Only the Paranoid Survive: How to Exploit the Crisis Points that Challenge Every Company and Career* (Doubleday, New York).

Hall, Mark, 2001, IBM roils Linux waters, *Computerworld*, Oct. 29, pp. 42-43.

Hess, Robert, 1995, AIX, MachTen, Linux to fill PPC Unix gap, *MacWEEK*, May 29, p. 16.

Katz, Michael L., Shapiro, Carl, 1985, Network Externalities, Competition, and Compatibility, *American Economic Review* 75 (3), 424-440.

Katz, Michael L., Shapiro, Carl, 1986, Technology Adoption in the Presence of Network Externalities, *Journal of Political Economy* 94 (4), 822-841.

Kim, Eugene Eric, 1999, The Joy of Unix, *Linux Magazine*, November.

Kogut, Bruce, Metiu, Anca, 2001, Open-source software development and distributed innovation, *Oxford Review of Economic Policy* 17 (2), 248-264.

Kraemer, Kenneth L., Dedrick, Jason, Yamashiro, Sandra, 2000, Refining and extending the business model with information technology: Dell Computer Corporation, *The Information Society* 16 (1), 5-21.

Langlois, Richard, 1992, External economies and economic progress: The case of the microcomputer industry, *Business History Review* 66 (1), 1-50.

Langlois, Richard, Robertson, Paul, 1995, *Firms, Markets and Economic Change* (Routledge, London).

Leibovitch, Evan, 2001, Open source's black hole, www.zdnet.com, 2 May (accessed 9 Oct. 2001).

Liebowitz, Stan J., Margolis, Stephen E., 1999, *Winners, Losers and Microsoft: Competition and Antitrust in High Technology* (Independent Institute, Oakland, Calif.).

Loukides, Mike, 1999, Some Thoughts on the Sun Community Source License, java.oreilly.com (accessed 30 Dec. 2001).

MacCormack, Alan, Herman, Kerry, 1999, Red Hat and the Linux Revolution, Harvard Business School case #9-600-009, Boston, Mass.

McKay, Niall, 1998, IBM bundles Apache into its WebSphere, *InfoWorld* June 22, p. 3.

McKusick, Kirk, 1999, Twenty Years of Berkeley Unix, in: Chris DiBona, Sam Ockman and Mark Stone (Editors), *Open Sources: Voices from the Open Source Revolution* (O'Reilly, Sebastopol, Calif.) pp. 31-46.

Moltzen, Edward F., Burke, Steven, 1998, Apache Server ported to NT, *Computer Reseller News*, 16 November, pp. 5,10.

Moore, Geoffrey A, 1991, *Crossing the chasm: marketing and selling technology products to mainstream customers* (HarperBusiness, New York).

Morris, Charles R., Ferguson, Charles H., 93, How Architecture Wins Technology Wars, *Harvard Business Review* 71 (2), 86-96.

Moschella, David C., 1997, *Waves of power: dynamics of global technology leadership, 1964-2010* (AMACOM, New York).

Mowery, David C. (Editor), 1996, *The International Computer Software Industry: A Comparative Study of Industry Evolution and Structure* (Oxford University Press, New York).

Nokia, 2001, Nokia to license a mobile terminal software platform and client components to handset vendors, Press release, 13 November, press.nokia.com (accessed 8 April 2002).

Raymond, Eric S., 1999, *The cathedral and the bazaar: musings on Linux and open source by an accidental revolutionary* (O'Reilly, Cambridge, Mass.).

Rogers, Everett M., 1995, *Diffusion of innovations*, 4th ed. (Free Press, New York).

Rosenberg, Donald K., 2000. *Open Source: the unauthorized white papers* (M&T Books, Foster City, Calif.).

Scannell, Ed, Gardner, Dana, 1999, Linux meets Netfinity, *InfoWorld*, 2 August, p. 1.

Shankland, Stephen, 1999, Apple modifies open source license, CNET News.com news.cnet.com 20 April (accessed 12 June 2001).

Shankland, Stephen, 2002, Mission impossible at IBM? CNET News.com news.cnet.com 30 April (accessed 18 Dec. 2002).

Stallman, Richard, 1999, The GNU Operating System and the Free Software Movement, in: Chris DiBona, Sam Ockman and Mark Stone (Editors), *Open Sources: Voices from the Open Source Revolution* (O'Reilly, Sebastopol, Calif.) pp. 53-70.

Stallman, Richard, 2001, as interviewed in: *Revolution OS*, 35mm film, J.T.S. Moore, director, Wonderview Productions LLC.

Sun Microsystems, 2001, OpenOffice.org FAQ, www.sun.com (accessed 30 Dec. 2001).

Taylor, Maureen, 1996, Scott Says...'Kick Butt and Have Fun': A candid interview with Scott McNealy, Sun's CEO, www.sun.com (accessed 30 Dec. 2001).

Teece, David, 1986, Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy, *Research Policy* 15 (6), 285-305.

Torvalds, Linus, Diamond, David, 2001, *Just for fun: the story of an accidental revolutionary* (HarperBusiness, New York).

Varhol, Peter D., 1994, Trends in operating system design: An Interview with Linus Torvalds, *Dr. Dobb's Journal* May, p. 18.

von Burg, Urs, 2001, *The Triumph of Ethernet: Technological Communities and the Battle for the LAN Standard* (Stanford University Press, Stanford, Calif.).

von Hippel, Eric, 1976, The Dominant Role of Users in the Scientific Instrument Innovation Process, *Research Policy* 5 (3), 212-39.

Wayner, Peter, 2000, *Free for all: how Linux and the free software movement undercut the high-tech titans* (Harper Business, New York).

West, Joel, 2000, A Comparison of PC Standard Switching Decisions by U.S. and Japanese Computer Users, Ph.D. dissertation, University of California, Irvine, Irvine, Calif.

West, Joel, 2003, The fall of a Silicon Valley icon: Was Apple really Betamax redux? Forthcoming in: Richard Bettis (Editor), *Strategy in Transition* (Wiley, New York).

West, Joel, Dedrick, Jason, 2000, Innovation and Control in Standards Architectures: The Rise and Fall of Japan's PC-98, *Information Systems Research* 11 (2), 197-216.

West, Joel, Dedrick, Jason, 2001, Open Source Standardization: The Rise of Linux in the Network Era, *Knowledge, Technology and Policy* 14 (2), 88-112.

Wilcox, Joe, 2000, IBM to spend $1 billion on Linux in 2001, CNET News.com news.cnet.com 12 December (accessed 15 Dec. 2000).

Zachary, G. Pascal, 1991, Free for All: Richard Stallman Is Consumed by the Fight To End Copyrighting of Software, *Wall Street Journal*, 20 May, p. R23.

Tables and Figures

Category	Firm	Platform	Released
Mainframe	IBM	IBM 360 (370,390)	1964
Minicomputer	DEC	VAX/VMS (OpenVMS)	1977
	IBM	AS/400	1988
	†*AT&T->OSF*	Unix	1980§
Workstation	Apollo	Domain¶	1980
	Sun	Sun OS (Solaris)	1982
8-bit PC	Apple	Apple II¶	1977
	†*Digital Research*	CP/M¶	1976§
16+ bit PC	IBM	IBM PC	1981
	†*Microsoft*	Windows	1990§
	NEC	PC-98¶	1983
	Apple	Macintosh	1984
Personal Digital Assistant	Palm	Pilot	1996
	†*Microsoft*	Windows CE	1996

† OS vendor; otherwise, vendors are vertically integrated manufacturers
§ Widespread commercial release
¶ Discontinued

Table 1: Major 20th century computing platforms

Product							
Vendor	IBM	DEC	DEC	Sun	Apple	IBM	Compaq
Product	S/390	VaxStation	VaxStation	SparcStation	Macintosh	PS/2	DeskPro
Segment	Mainframe	Workstation	Workstation	Workstation	PC	PC	PC
Platform							
Applications	*own*	*own*	*own*	*own*	*own*	*own*	*3rd party*
	3rd party	*3rd party*	*3rd party*	*3rd party*	*3rd party*	*3rd party*	
OS	OS/390	VMS	Unix†	Unix†	Mac OS	MS-DOS	MS-DOS
	(*own*)	(*own*)	(*3rd party*)	(*3rd party*)	(*own*)	(*3rd party*)	(*3rd party*)
CPU	ES9000	CVAX	CVAX	Sparc	68030	386	386
	(*own)*	(*own)*	(*own)*	(*own*)	(*3rd party)*	(*3rd party*)	(*3rd party*)

† Licensed from a 3rd party supplier but with proprietary extensions

Table 2: Ownership of achitectural layers for representative computer platforms, 1990

	Platform strategy	Sponsor	Multiple hardware vendors	Multiple OS vendors	Source Licensing	Published APIs for 3rd party s.w	API Control	Products
closed	Vertically integrated proprietary	Hardware vendors	no	no	no	yes	hardware vendor	IBM S/360, DEC VAX, Macintosh
	Horizontal proprietary	Microsoft	yes	no	no	yes	software vendor	Windows 3.0
	Unix	AT&T	yes	no	yes	yes	software vendor	System V
open	Open Systems	*Consortia*	yes	yes	yes	yes	consortium	OSF, X/Open

Table 3: Representative platform IPR strategies, 1990

Date	Industry	Apple	IBM	Sun
May 1995				Pre-release Java posted for free Internet download
October 1995		Starts working on MkLinux for Power Macintosh		
March 1996				Licenses Java to Microsoft
May 1996		Releases MkLinux, gives away 20,000+ CDs to ISVs		
October 1996				Sues Microsoft over “polluted” Java
December 1996		Buys NeXT to adapt Mach/BSD-based OS		Announces “100% Pure Java” initiative
June 1998			Announces use of Apache across entire server line	
July 1998	Torvalds on cover of *Forbes*	Ends work on MkLinux		
September 1998	Microsoft lists Linux as a threat in SEC filing		Announces DB2 will support Linux	
November 1998	Microsoft “Halloween” memo stirs controversy			Abandons ISO/IEC JTC1 Java standards effort
December 1998			German IBM engineers port Linux to S/390 mainframe	Intro Sun Community Source License, plans to release Java source code
December 1998			Announces open source for Jikes Java compiler	Announces support for Linux on UltraSparc CPUs
February 1999				Java 2 source code posted under SCSL
March 1999	IBM, Compaq, Oracle and Novell invest in Red Hat	Announces Darwin open source OS and Apple Public Source License	Supports Linux on work-stations and PC servers	
May 1999		Releases Darwin source code on Apple web server	Certifies 4 Linux versions for PC servers	
July 1999			Releases DB2 for Linux	
August 1999	After IPO, Red Hat reaches $4 billion market cap			Buys StarOffice, reveals plans to release source
September 1999			Launches developerWorks open source site	Releases SPARC chip design under SCSL
October 1999	Microsoft posts "Linux Myths" page, attacks Linux			Announces it will release Solaris under SCSL
December 1999				Abandons ECMA standards process on Java; Resells Red Hat Linux
February 2000			Posts its Linux for S/390 for free download	
August 2000	Open Source Development Lab funded		Releases tools under IBM Public License	Replaces own GUI with open source GNOME
September 2000		Releases public beta of Darwin-based OS X		Purchases Cobalt, maker of Linux server appliances
October 2000				Posts StarOffice source under Lesser GPL license
November 2000				Red Hat drops Linux support for Sun systems
December 2000			Announces IBM will spend $1 billion on Linux in 2001	Releases Solaris source under SCSL license
January 2001				Settles Java lawsuit with Microsoft
March 2001	Microsoft launches “Shared Source” initiative	Ships Mac OS X, based on Darwin and FreeBSD	Launches “Peace, Love and Linux” ad campaign	

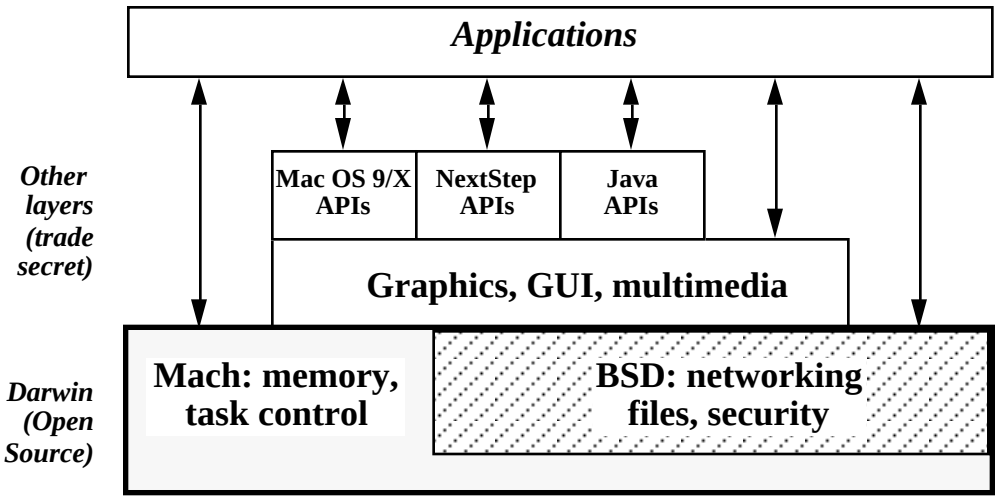
June 2001	Microsoft CEO Ballmer says “Linux is a cancer”	Hires FreeBSD co-founder		
November 2001			Launches Eclipse open source consortium	
April 2002	HP wins Linux-based supercomputer contract with U.S. Dept. of Energy	With non-profit Internet Software Consortium, co-sponsors OpenDarwin.org		
May 2002				Sells StarOffice for $76
September 2002			Red Hat Linux expands support to all IBM servers	Introduces LX series Linux-based PC servers

Table 4: Open Source Platform Milestones, 1995-2002

	Earlier Platforms		***Revised Platforms***	
Category	Hardware	Operating System	Rebranding	Operating System†
mainframe	IBM S/390	OS/390	zSeries	Linux
minicomputer	AS/400	OS/400	iSeries	Linux
workstation	RS/6000	AIX	pSeries	Linux
server PC	Netfinity	Windows NT Windows 2000	xSeries	Linux
desktop/laptop PC	NetVista ThinkPad	Windows 95/98/ME	*n/a*	*n/a*
PDA	WorkPad	Palm OS	*n/a*	*n/a*

† Traditional OS also available on rebranded platforms

Table 5: IBM’s supported platforms before and after Linux adoption



Source: Apple Computer (2000)

Figure 1: Architectural layers for Apple’s Mac OS X platform

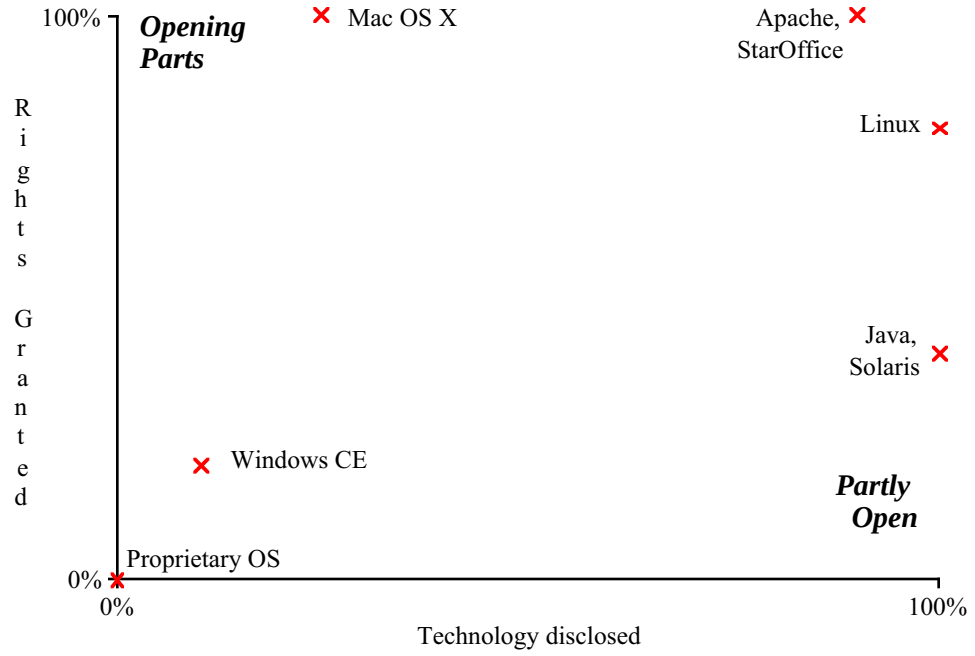


Figure 2: User rights under open and quasi-open source licenses