17 Shared Source: The Microsoft Perspective

Jason Matusow

Within the software industry, the debate continues about the roles of open source, free, commercial, and noncommercial software. The reality is that there are far more commonalities than differences. Where differences do exist, though, it is helpful to examine their implications for businesses, individuals, academic institutions, and government organizations.

The dialogue surrounding the different approaches to software development and distribution covers a vast array of issues, including consumer flexibility, cost versus value, economic opportunity, intellectual property (IP) rights associated with software, industry standards, security, privacy, business and licensing models, and more. These broad themes are woven with source code as a common thread. On the surface, source code has always been the exclusive domain of the programmer. But underlying the engineering aspects of source code generation and modification are some fundamental questions regarding the future of software innovation. Therefore, the debate goes on.

The odd thing about source code is that many speak about it, but few are able or willing to work with it. In 2002, with the goal of establishing clarity and context, Microsoft undertook private research regarding source code access for the software being used by businesses and governments.[1] Conventional wisdom might have predicted that we would find corporate IT professionals scouring source code in their daily work. Instead, we found that approximately 95 percent of organizations do not look at the source code of the operating systems serving as the core of their technology infrastructure. In addition, we found that while approximately 5 percent do look at the source code, less than 1 percent will modify it. As one looks at increasingly smaller organizations, the practice of accessing and modifying source code further drops.

The barrier to entry for understanding complex source code is significant. Although there are millions of software developers in the world

today, they still represent a small fraction of the total population of those using computers. Furthermore, there is an uneven distribution of development skills among programmers, so the community looking at highly complex code is smaller still. For most organizations, the cost and relative benefit of employing highly skilled developers is prohibitive, especially considering the abundance of quality packaged software.[2]

Even so, organizations stated that the opportunity to access operating system source code is important to them.[3] The majority of companies and governments supported the option of seeing source code. Simply put, transparency increases trust.

This suggests that to most people having the *option* of doing something is of far greater importance than actually doing it. For example, look at the idea behind government-mandated full disclosure of the financial information of publicly traded companies. Even though these statements are public, they are extremely complicated and require a thorough understanding of finance to truly gain insight into the health of a given firm. The vast majority of private investors are dependent on a relatively small community of professionals to interpret the numbers and provide guidance. The option of viewing the numbers is broadly available, and trust is therefore established through the availability of transparency. For most, though, it is an option that will never be exercised.

Transfer the private investor scenario to the typical users of today's operating systems and the situation looks much the same. Most organizations or individuals have no intention of going under the hood of their operating system to tinker with source code.[4] Organizations and average consumers depend heavily on commercial vendors to provide the expected levels of quality and support. This is where commercial software providers deliver value in the products they build and sell.[5]

Over the past few years, I have been running the Microsoft Shared Source Initiative. Through this initiative, we are making various types of Microsoft source code available to customers, governments, partners, and competitors worldwide. Some of our source code, such as that for Windows, permits reference use only (meaning that no modifications can be made), while our other programs, covering technologies such as Windows CE.NET, allow for modifications and redistribution of that source code.[6]

Through my work on Shared Source, I have had countless conversations with individuals and organizations about the role of source code in meeting their particular needs. Even though we have delivered source code to more than a million engineers, it is a tiny percentage of the total devel-

oper population working with Microsoft technologies. Our practical experience on a global scale confirms the relationship between the operational and peace-of-mind needs described earlier. Again, the factors of transparency, choice, trust, and need all play a role in our approach to the licensing of our source code.

Our approach to this issue is based on three simple ideas. First, our customers want source access both for its technical benefits, and because transparency increases trust. Second, there is no uniform way for Microsoft to provide source access that covers all business and licensing needs across all product offerings. Third, customers will be more successful with the source code if solid tools and information are provided along with the technology. Under these basic assumptions, Microsoft has been developing the Shared Source approach. Shared Source is not open source; rather, it is the means for a company that directly commercializes software to provide source code access without weakening its competitive differentiators or business model. Microsoft recognizes the benefits of the open source model, yet also understands that it is not necessarily a model that will work for everyone.

The goals of this chapter are twofold. First, to place the Shared Source Initiative and commercial software into the broader context of the ongoing source licensing debate. Second, to provide insight into how Microsoft has approached the licensing of its core intellectual property assets.

A Natural Move to the Middle

In 2000 and 2001, there appeared to be clear delineation among those involved in the source licensing debate. Microsoft was seen to be a polarizing factor as a continuum of positions was established, with the traditional intellectual property holders at one end and those opposed to software as commercial property at the other. Individuals and organizations advocating open source software deliberately positioned themselves as an alternative to Microsoft's practices or as active opponents of Microsoft. Now in 2004, as everyone deals with the aftershocks of the dot-com era, a wave of practicality has washed over businesses and individuals alike.

Open source software (OSS) itself, as a classification of software, has bifurcated into commercial and noncommercial segments. For many, the most interesting OSS work going on today falls into the fully commercial category, since significant dollars, resources, and technology are coming

from those seeking to use OSS as the basis for strategic business purposes (see table 17.1).

Careful observation of the commercial software community shows a consolidation of source licensing practices by a majority of the most significant players. In today's marketplace, software development, licensing, and business strategies fall under a blend of community and commercial models. Few software companies are left that can properly call themselves either purely OSS (in the sense of OSS as a community-driven, not-for-profit exercise) or purely commercial.

For the sake of this discussion, let's draw a line of distinction between noncommercial and commercial software. The merits of both may be observed in the software ecosystem that has developed over the past 30 years, as discussed later in this chapter.

Noncommercial software may be roughly grouped into three categories:

- *Research:* Government and academic researchers who produce technologies designed to move the general state of the art forward.
- *Teaching and learning:* Professors, students, and self-educators who work with, and learn from, software that is available free of charge and have no intention of commercializing the software generated in the learning process.
- *Community development and problem solving:* Hobbyists and professional developers who produce software with no intention of commercialization; this software may be meant to replace existing commercial options or to solve problems vendors have not addressed.

Commercial software may be roughly grouped into two categories:

- *Direct commercialization:* Those who use the product of community and/or corporate development as a mechanism for generating a direct revenue stream.
- *Indirect commercialization:* Those who use the product of community and/or corporate development to facilitate the success of another product or service for the generation of revenue.

It is worth noting here that the concepts of noncommercial and commercial software have nothing to do with the availability of source code. If a long-standing commercial software vendor provides source code of a given product, this does not change the fact that the software is commercial in nature. At the same time, if a piece of software is produced and maintained as communal, noncommercial software, there is no reason a commercial entity may not make use of it without altering its standing as noncommercial software.

Table 17.1
Software development strategies

Direct commercialization	Community development	Red Hat Inc.'s distribution of Linux is a combination of community-built software through the Free and Open Source models and corporate-funded professional development contributions. The pricing of its Premium Editions, its certification practices for hardware and applications, and its support policies are all mechanisms to directly commercialize the operating system.
		Apple Computer Inc. has combined community software with commercial software to create the OS X operating system. The company is directly commercializing the software while utilizing community-developed code.
	Corporate development	Microsoft has built the Windows product using corporate development resources. The product is directly commercialized through the licensing of the binary version of the product. The source code is now available to a limited community through the Shared Source Initiative.
		CollabNet Inc. has built a proprietary tool that is directly commercialized through the licensing of the binary version of the product and through associated services. The product facilitates the use of the OSS development model, which could be used to create noncommercial software.
Indirect commercialization	Community development	IBM Corp. has heavily participated in the community-based development of the Apache Web server. While IBM is not directly commercializing the Apache server, it is driving the return on investment revenue stream through the sale of the WebSphere product. RealNetworks Inc. released significant

Table 17.1
(continued)

		segments of its Helix product source code, which was originally commercially developed. The goal of community development around the Helix product set is to generate a larger market for other revenue-generating products.
	Corporate development	Adobe Systems Inc.'s Acrobat Reader is a product of corporate development and of closely held intellectual property. Reader is downloadable at no cost to drive the sale of the full Acrobat product. (Adobe Systems Inc. does provide the file format specification for .pdf files, but they are not releasing the source code to their implementation. More information may be found at http://www.adobe.com.)
		Driver development kits (DDKs) and software development kits (SDKs) are provided by all commercial operating system vendors (examples include Novell Inc.'s NDK and Microsoft's DDK). These developer tools often contain sample source code that can be modified and redistributed, yet the kits themselves are provided at no cost to developers. There is no direct commercial value to the DDKs or SDKs themselves; rather, they create opportunity for others to build software and hardware for that platform.

Table 17.1 maps the commercial categories listed previously to examples from the software industry. Many of the companies in the table closely associate themselves with the OSS movement, yet they are clearly commercial enterprises. Some of those listed have no direct affiliation with the concepts of OSS, yet their behavior can be instructive, specifically, their approach to the distribution of software.

A common misperception about software developed under the open source model is that a random group of distributed developers is creating the software being adopted by businesses. Although this is true for some smaller projects, the reality is that professional corporate teams or highly structured not-for-profit organizations are driving the production, testing, distribution, and support of the majority of the key OSS technologies. The concepts behind source code access are not the determining factors as to whether the software is commercial. Source code access plays a role in both commercial and noncommercial environments.

Source code access issues unquestionably affect the future of innovation in the industry. The move to the middle outlined earlier is a result of the influence of these issues on the industry to date.

The Software Ecosystem

At the core of software evolution is the interaction among government, academic, and private research. These relationships represent an integrated, natural ecosystem. Though these organisms exist independently and conduct independent "development," there are clear areas of interdependency that yield dramatically greater results for the whole.

This ecosystem has been at the heart of the ongoing cycle of sustained innovation that has made information technology one of the most dynamic industries in the economy.[7] The blending of differing development, licensing, and business models has been the central factor for success.

Governments and universities undertake basic research and share this knowledge with the public.[8] In turn, companies in the private sector use some of these technologies in combination with their even greater ongoing investment in research and development[9] to create commercial products, while also contributing to the work of common standards bodies. Their success leads to greater employment and tax revenues, as well as additional funding for academic research projects.[10]

The concepts associated with the software ecosystem are not unique to discussions of source code access. Take aviation, for example. Although the

vast majority of us have little everyday use for an F-15 Eagle jet fighter, government and academic research and development behind the fighter for everything from metallurgy to heads-up displays have benefited the production and operation of commercial airplanes.

For a more IT-centric example, consider TCP/IP. Born as a government research project, it matured in academia under the OSS development model and evolved into an open industry standard. After that, it was further refined and brought into the computing mainstream via proprietary implementations by commercial software companies such as Novell, Apple, IBM, and Microsoft.

Microsoft's Windows operating system, on the other hand, was developed privately and for profit. But the product includes many components born of government and academically funded work and contains implementations of dozens of open industry standards. Furthermore, the publication of thousands of application programming interfaces created business opportunities for tens of thousands of software businesses and has resulted in innumerable custom applications that address individual needs.

So where will this line of reasoning take us? If the past is any indication, the future of software will not be the result of the dominance of a single development, licensing, or business model. Future innovation will not come solely from government, private industry, or a loose coalition of individuals acting in the best interests of society at large. The continued health of the cycle of sustained innovation—the fruits of which we have enjoyed for three decades—will depend entirely on the continued melding of approaches and technologies. In the end, the consumers of software, both custom and packaged, will be the beneficiaries, particularly as natural market forces continue to shape the actions and results of corporations and individuals alike.

Striking a Balance

Given the ongoing move to the middle of software vendors and the effects of the software ecosystem, the question for Microsoft has been how to find the proper balance between greater transparency, sustainable business, and innovation investment.

OSS is clearly having an effect on how software companies think about and treat their intellectual property assets.[11] The sharing of source code, while beneficial in many ways, also presents challenges to the accepted concepts of the commercialization of software and competitive differenti-

ation. The modus operandi for most software companies has been to closely protect IP assets in software products to maintain uniqueness and competitiveness in the market. Trade secret law, which covers many aspects of software that are obscured by the compilation process, has played a pivotal role in the IP protection strategy of most commercial software companies. Historically, this protection has been maintained through either binary-only distribution or source code distribution under nondisclosure agreements. Yet OSS and other source-sharing models are moving organizations to seek a balance between IP protection (particularly with respect to protection of trade secrets) and customer/partner benefit.

Arguably, intellectual property rights have become more important as the desirability and functionality of transparency increases. The creative use and combination of all four forms of IP protection are paving the way for further source code sharing by allowing companies to selectively ratchet back trade secret protection.[12]

This is not to say that it always makes sense for companies to provide the source code to their software products, thereby limiting or destroying their trade secrets. Commercial software companies balance perceived customer needs and desires with a host of other business concerns. For instance, many investors demand that companies protect their assets through all means possible so as to protect the future returns on investment and ensure a healthy revenue stream. Anyone who has ever gone through the process of attempting to raise capital for a software business can attest to this. In some situations, it may be that trade secrets in a software product are essential to preserving the market advantage of that product. Accordingly, a company would be unwilling to make the source code to that product available.

Most successful software businesses reinvest material amounts of their gross incomes into research and development. Microsoft is now investing approximately $5 billion annually, or approximately 15 percent of gross revenues, in our future.[13] So where does this leave us? How do you balance the obvious benefits of source code transparency and flexibility for developers against the software business reality of protection of assets and the need for healthy sources of revenue?

Each software company must decide for itself which path to take. For Microsoft, it was clear that customers, partners, and governments were eager for us to move toward transparency and flexibility. At the same time, we had to take that step with an eye to the other side of the equation as well. Through this process, we created the Shared Source Initiative.

The Shared Source Initiative

Microsoft is sharing source code with customers, partners, and governments globally. We have released source programs delivering well over 100 million lines of source code. The Shared Source Initiative evolved as we sought to both address customer and partner requests to have greater access to source code and look carefully at the benefits and potential pitfalls of the OSS and Free Software approaches. We then selectively applied lessons learned from those approaches and our existing business model to better meet customers' needs.

Shared Source is a framework, not a license.[14] Any commercial software company needs to analyze the interplay among the elements of development models, licensing, and business models to establish a successful strategy whereby source code may be shared or opened in a way that benefits customers without jeopardizing the company's ability to remain in business. Microsoft's licensing approach ranges from reference-only grants (where licensees may review Microsoft source code for the purposes of reference and debugging, but are not granted modification or redistribution rights) to broad grants that allow licensees to review, modify, redistribute, and sell works with no royalties paid to Microsoft.

There are now hundreds of thousands of developers with Microsoft source code. We have taken what is arguably the most commercially valuable intellectual property in the software industry and made it available to thousands of organizations in more than 60 countries.[15] Shared Source programs now deliver source code for Windows, Windows CE.NET, Visual Studio.NET, C#/CLI, ASP.NET, and Passport technologies. Over time, we will continue to evaluate source code as a feature of our products and also how our customers and partners may best use the source code.

One of the most common misperceptions of the Shared Source model is that it is limited to "look but don't touch" and a single license. In fact, Shared Source covers four key concepts:

- *Support existing customers:* Provide source access for existing customers to facilitate product support, deployments, security testing, and custom application development.
- *Generate new development:* Provide instructional source code through samples and core components for the facilitation of new development projects.
- *Augment teaching and research:* Provide source code and documentation for use in classrooms and textbook publishing and as a basis for advanced research.

▪ *Promote partner opportunity:* Provide licensing structure and source code to encourage mutually advantageous new business opportunities for partners.

At the time of the writing of this chapter, seven Microsoft product groups are providing source code with some ability to create derivative works of the source code.[16] Three of the groups are placing source code in the community's hands with rights to create derivative works and distribute them commercially, meaning that a programmer can get the code, modify it, and redistribute it under a traditional commercial binary license for profit—never paying Microsoft a dime. All the current source code programs from Microsoft are provided at no cost.[17]

Building a Shared Source Program

Microsoft has applied significant resources to establish the various Shared Source programs. Every source release depends on a series of decisions made to deliver the proper balance between customer and business benefits. We have also invested in engineering resources to deliver augmented tools and documentation that increase the value of source code access.

Table 17.2 provides a small sample of the questions and process Microsoft works through when establishing a source release. This is by no means a complete analysis tool, but rather a sample of the decision-making process for a commercial software provider considering a source-sharing program.

A well-designed source release should accomplish a few key goals:

▪ Provide educational insight into the product or project being shared. Value is derived for many simply through availability and analysis rather than modification of source code.
▪ Deliver related tools, documentation, and support to increase the value to the individual working with the source code, particularly in programs with derivative rights associated with them.
▪ Establish clear feedback mechanisms to facilitate the improvement of the code base.
▪ Identify the community that most benefits from access to the source code.
▪ Define a set of rights that protects the creator of the source code and those working with the source code.

Table 17.2
Shared source consideration

	Questions	Considerations
Determining objectives	What community are you planning to serve with this source code?	Not all source releases have to be global and available to the general public. Working with gated communities, large customers, key partners, or particular government agencies might be more appropriate for some situations.
	What is the benefit of this source code to these organizations and individuals?	Source code does not address all IT concerns. Understanding how the source will be beneficial is a critical factor in determining granted rights and delivery mechanisms.
	How many people will have the source code and how will you interact with them?	Broad-reach programs may have significant resource requirements for source delivery and/or community participation. This goes beyond logistical concerns such as download capacity. The amount of engineering participation, feedback processing, and continued investment, among other elements, must be considered.
	What geographies will be eligible for source access?	Aside from the more obvious concerns about the localization of documentation, there are significant legal differences to be considered from country to country. The treatment of IP issues varies greatly and you should seek legal counsel on this issue. (This concern is universal for any source model—OSS, Shared Source, or otherwise. For example, many of the most popular OSS licenses are based on assumptions of the U.S. copyright system. Because code is used globally, the legal standards applied to licenses vary greatly.)

Table 17.2
(continued)

	Questions	Considerations
Source management	What source are you planning to share?	Just as the community you plan to share source code with is not necessarily 100 percent of the population, the source code you share does not have to represent 100 percent of a product. Certain components of a product are extremely valuable, are licensed to you by a third party under terms that prohibit disclosure of source code, or are subject to government export restrictions.
	Do you have rights to share the targeted source base?	Commercial software often contains components that originated elsewhere and are being reused or licensed within a larger product. The greater the number of copyright holders there are for a given piece of software, the greater the complexity involved in source-sharing of commercial products.
	Have you cleaned the code for public consumption and thought about the quality of comments?	Within source code, developers place comments to provide insight into their thought processes. There can be significant value in augmenting comments within particularly complex code segments. Unfortunately, there is often colorful language in source code, and public consumption should be taken into account.
	Do you have a source management strategy in place for bug fixes and future releases related to the public source program?	Most successful software engineering projects have mature source management processes in place. If you are taking a previously nonshared code base and moving it out to a broader set of developers, you need to establish a process for delivering

Table 17.2
(continued)

	Questions	Considerations
		ongoing inhouse engineering work to the community that now has the source code.
	How will you handle incoming suggestions or code fixes?	Along the same lines as new inhouse code delivery, you need to establish a process for receiving suggestions, bug fixes, and new features from the community that has access to the code. You might also want to consider the legal ramifications and risks associated with incorporating incoming suggestions or code fixes into your code base.
Licensing	What rights do you plan to give for viewing, debugging, modifying, and distributing?	Although the input of attorneys is important in establishing licensing rights for source code, the far more important voice is that of your customers and partners. The driving factor must be how they will benefit most from the source code. Fiduciary responsibility to investors regarding the protection of IP is critical as well, but that thinking should be applied secondarily. A successful source license program establishes a balance between these factors to the benefit of all involved.
	If you grant derivative rights, can the redistribution be commercial?	Microsoft has opted to implement two types of derivative work license approaches to differentiate business goals within our source licensing programs. The commercialization of derivative works is a key focal point for commercial software vendors releasing source code.

Table 17.2
(continued)

	Questions	Considerations
Fulfillment	What is the delivery mechanism for the source code? Will it simply be a package of source code or will there be added value through a delivery tool?	There is a wide range of source delivery options. The OSS community has a number of sites such as VA Software Corp.'s SourceForge for the delivery of source code and the management of projects. You may choose to host your own environment, as Microsoft has done with its GotDotNet WorkSpaces project. Microsoft also built a secure Web infrastructure for the delivery of Windows source code: MSDN Code Center Premium. Other groups within Microsoft have opted for simple Web downloads of source files, leaving the choice of toolset and engineering environment up to the individual developer. Determining the size of the community to be involved and the amount of source code included in the release is an important factor in choosing a delivery mechanism.
	How will you engage the community of individuals and organizations who have the source code?	Although a community of developers may spontaneously form around a given source base, successful programs will likely include involvement from the engineers who built the code. Furthermore, strong project management helps keep community involvement productive.
	Have you produced additional support documentation for the source base?	The creation of good software documentation has proven to be one of the most expensive and difficult problems in the industry. The more information provided to developers working with a given source base, the better. Although this is not mandatory, it will certainly improve the quality of the source-sharing program as a whole.

Lessons Learned and a Look Ahead

The most fundamental lesson we have learned through the Shared Source process is that source code is a product feature. For many, it is a feature that will never be used, but an option that is good to have. Our customers and partners who have source code, and who are actively using it, tell us that it is invaluable to the way they use our products. Yet this number represents a minute fraction of the total number of individuals and organizations using our products.

Microsoft's Shared Source Initiative is only two years old, but we have been providing source code to academic institutions and OEMs for more than 12 years. Before 2001, our source sharing reached a limited audience and was less formal than it is today. We have been listening to our customers and are learning from OSS—that is the blend to which we aspire with this initiative. In many ways, Shared Source is still in its version 1.0 phase. The success of the programs to date has shown us the importance of expanding this initiative into other code bases.

Licensing of source code will continue to be a hot button for the industry for the foreseeable future. There are some basic questions about the role of IP in future innovation. At the heart of Shared Source is a belief that intellectual property, and the protection of that property, is behind the fundamental success of an ongoing cycle of sustained innovation. The dissemination of source code is exceedingly beneficial, but not to the exclusion of a successful software industry. Nor is it the panacea for all information technology concerns; multiple models will continue to coexist. It is one part of a much larger puzzle, and I for one am glad to be sitting at the table working on a few small pieces.

Notes

1. This private research involved more than 1,100 individuals in five countries representing business decision makers, IT professionals, and developers. The study was completed in April 2003.

2. This premise is based on an extremely simplified view of Ronald Coase's concept of transaction costs and how they influence organizational behavior. Coase was awarded the Nobel Prize in Economic Sciences in 1991 for his work regarding the significance of transaction costs and property rights for the institutional structure and functioning of the economy (http://coase.org, accessed May 20, 2003).

3. The same research in note 1 revealed that approximately 60 percent of respondents felt that having the option to view source code was critical to running software in a business setting.

4. This is equally true for Windows, Linux, Mac OS, Netware, OS/400, and other major commercial operating systems. The smallest of these represents millions of lines of source code (an operating system is more than just a kernel), and as they mature, complexity increases rather than decreases.

5. Clearly, the capability of community support cannot be underestimated. For years, there have been newsgroups and mailing lists where the community has provided support for commercial, open, free, and shareware software. When organizations are dealing with their mission-critical systems, however, they primarily seek professional support with service-level agreements to mitigate risk.

6. As of May 2003, Microsoft has programs in place for Windows, Windows CE.NET, Visual Studio.NET, C#/CLI, ASP.NET, and Passport. Under the Windows programs, only academic researchers are given rights to modify the source code. For all other source programs, modification and redistribution rights are granted. Please see http://www.microsoft.com/sharedsource/ for further details.

7. In the latter half of the 1990s alone, information-related industries, representing 8.3 percent of the U.S. economy, fueled approximately 30 percent of overall economic growth and at least half the acceleration in productivity rates (U.S. Department of Commerce, U.S. Government Working Group on Electronic Commerce, "Leadership for the New Millennium: Delivering on Digital Progress and Prosperity," Jan. 16, 2001).

8. In fact, U.S. federal agencies are required by law to encourage certain grant recipients and public contractors to patent the results of government-sponsored research, and universities are active in asserting research-related intellectual property rights. Since at least 1980, the U.S. government has pursued a vigorous policy of transferring the results of federally funded technology research to industry to promote innovation and commercialization. See the Bayh-Dole Act of 1980, the Stevenson-Wydler Technology Innovation Act of 1980, the Federal Technology Transfer Act of 1986, Executive Order 12591 of 1987, the National Technology Transfer and Advancement Act of 1995, and the Technology Transfer Commercialization Act of 2000.

9. From 1969 through 1994, U.S. high-tech R&D investment was $77.6 billion from the government and $262 billion from private industry (the National Science Foundation's Industrial Research and Development Information System (IRIS); accessed Oct. 11, 2002, at http://www.nsf.gov/sbe/srs/iris/start.htm.)

10. A good example of this is Google, Inc. Google was federally funded as one of 15 Stanford University Digital Libraries Initiative Phase 1 Projects. In 1996, the technology was disclosed to Stanford's Office of Technology Licensing (OTL). In 1998,

the OTL gave permission to Sergey Brin and Larry Page to establish a commercial entity based on the technology. Today, Google, Inc. is a successful firm generating revenue for both the company and Stanford University.

11. Within the source licensing industry debate, there are some who argue about the use of the term *intellectual property*. The use of it here covers the holistic concept of copyright, patent, trade secret, and trademark.

12. Recent cases involving both SuSE Linux and Red Hat have highlighted the importance of trademark in the open source business model.

13. It is not uncommon for software firms to reinvest between 15 and 30 percent of gross revenues in R&D.

14. http://www.opensource.org, visited Sept 8, 2004. There are 54 licenses that the Open Source Initiative has stated meet its criteria for being an "open source" license. As commercialization of OSS continues to expand, and as commercial software companies continue to push the limits of source sharing, there is likely to be a continued proliferation of source licenses as each organization and individual determines what terms it is most comfortable with for the distribution of its IP.

15. At this time, there is no comparable sharing of source code for flagship products in the software industry. Although many vendors provide top customers with sources upon request, few have put broad-reach programs in place. It is likely that this status will change over time as the positive effects of OSS, Shared Source, and other source delivery programs continue to be recognized.

16. The Windows Academic Shared Source and OEM licenses allow researchers to create temporary modifications of the Windows source code for research and testing purposes. All other constituency groups with Windows source access (enterprise customers, system integrators, and government agencies) have reference-only rights—meaning that they may view and debug, but may not create derivative works of the source code.

17. Due to the delay between writing and publishing of this document, specific details of the programs have been left out. If you would like further information on source availability from Microsoft, please visit http://www.microsoft.com/sharedsource/.