

Making Money with Open-Source Business Initiatives

Paul Benjamin Lowry

Brigham Young University, USA

Akshay Grover

Brigham Young University, USA

Chris Madsen

Brigham Young University, USA

Jeff Larkin

Brigham Young University, USA

William Robins

Brigham Young University, USA

INTRODUCTION

Open-source software (OSS) is software that can be used freely in the public domain but is often copyrighted by the original authors under an open-source license such as the GNU General Public License (GPL). Given its free nature, one might believe that OSS is inherently inferior to proprietary software, yet this often is not the case. Many OSS applications are superior or on par with their proprietary competitors (e.g., MySQL, Apache Server, Linux, and Star Office). OSS is a potentially disruptive technology (Christensen, 1997) because it is often cheaper, more reliable, simpler, and more convenient than proprietary software.

Because OSS can be of high quality and capable of performing mission-critical tasks, it is becoming common in industry; the majority of Web sites, for example, use Apache as the Web server. The deployment of OSS is proving to be a productive way to counter the licensing fees charged by proprietary software companies. An organized approach to distributing cost-effective OSS products is intensifying as companies such as RedHat and IBM co-brand OSS products to establish market presence.

From a business perspective, the entire OSS movement has been strategically anti-intuitive because it is based on software developers freely sharing source

code—an act that flies in the face of traditional proprietary models. This movement raises two questions this article aims to address: (1) why would individuals write software and share it freely? and (2) how can software firms make money from OSS? Before fully addressing these questions, this article examines the historical development of OSS.

OSS HISTORY

A strategic irony of the software industry is that its foundation rests primarily on OSS principles. Software development in the 1960s and 1970s was steered primarily by government and academia. Software developers working in the field at the time considered it a normal part of their research culture to exchange, modify, and build on one another's software (Von Krogh, 2003). Richard Stallman, a professor and programmer at MIT, was a strong advocate and contributor to this culture of open, collaborative software development. Despite Professor Stallman's influence, MIT eventually stopped exchanging sourcecode with other universities to increase its research funding through proprietary software licensing. Offended by MIT's decision to limit code sharing, Professor Stallman founded the Free Software Foundation in 1985 and developed the General Public

License (GPL) to preserve free code sharing (Bretthauer, 2002).

In the formative years of the software industry, Stallman’s free software movement grew slowly; in the early 1990s, however, the concept of code sharing grew more rapidly for a couple of reasons. First, “free software” was renamed “OSS,” a name that spread rapidly throughout the code-sharing community (Fitzgerald & Feller, 2001). Second, the OSS movement received a boost from the advent of the World Wide Web (WWW). The Web provided an opportunity for Internet users to quickly and conveniently share their code.

WHY DEVELOPERS WRITE OSS

The majority of OSS software developers fall into one of the following three categories: freelancers, software enthusiasts, or professionals. Freelancers enjoy the challenges associated with developing OSS and providing services to the OSS community to further their own careers. When freelancers create modules of code, they often include their contact information inside the modules (Lerner & Tirole, 2002). This allows businesses to contact the developers to request their future services.

Software enthusiasts are people who contribute to OSS simply out of the joy and challenge of doing so, with little regard for professional advancement. Enthusiasts are often university students who want to participate in the development of free software and who receive personal gratification from participating in real-world OSS development projects and gaining the respect of the OSS community.

Even though OSS is “free” software, many companies hire professional developers to work on improving OSS code. RedHat, a Linux support company, hires developers to fix bugs in OSS code and to create new applications (Lerner & Tirole, 2002). Other companies hire OSS developers because their systems run OSS applications and they need developers to customize the code for specific business purposes. Table 1 summarizes the different motivations for joining OSS projects and shows them on a spectrum of intrinsic and extrinsic motivations.


SOFTWARE DEVELOPMENT ECONOMICS

Proprietary software

The strategic motivation behind the creation of proprietary software is to set up high switching costs for consumers. For such companies their developers’ resulting source code becomes the company’s intellectual property and an unshared key company asset. Once customers purchase proprietary software, they must pay for updates continually to keep the software current, and often to receive full customer support (DeLong & Froomkin, 2000). Most customers will pay these fees because of the lock in that occurs from the often costly prohibitive tradeoff of implementing a completely new system.

Microsoft is an example of a company that has succeeded in proprietary software, largely because they have a focused strategy of selling complementary products and services to their installed base of Windows users (Shapiro & Varian, 1998): Offering

Table 1. Developer motivations

Enthusiast	Freelancer	Professional
<ul style="list-style-type: none"> • Learn • Earn respect 	<ul style="list-style-type: none"> • Challenge of developing code • Receive future job opportunities 	<ul style="list-style-type: none"> • Programming income • Customize OSS
<div style="display: flex; align-items: center; justify-content: center;"> Intrinsic  Extrinsic </div>		

complementary goods that run on Windows (e.g., Office) increases profitability and successfully enhances the buyer relationship while encouraging customer entrenchment.

Proprietary software development is rigidly structured. Development begins with an end product in mind, and the new product often integrates with other products the company is currently selling. Project leaders create development plans, set deadlines, and coordinate teams to develop modules of the new software product. Successful proprietary software companies are also able to develop new technologies in exceptionally short time frames and to place their products in the market faster than their competitors. Products that meet the strict demands of end users succeed and increase customer satisfaction.

The downside of proprietary software development is that it comes at a tremendous internal cost (Lederer & Prasad, 1993); meanwhile, the industry is experiencing increasing pressures to decrease costs. Companies must invest heavily in research and development (R&D), human capital, information technology, marketing, brand development, and physical manufacturing of the products. They must continually innovate and develop updated versions of existing products, or create entirely new products. To compensate for these costs, proprietary software companies have high-priced products. Some software costs are so high that many businesses question whether the software is worth it.

OSS

The economics of OSS differ significantly in that OSS is developed in a loose marketplace structure. The development process begins when a developer presents an idea or identifies a need for an application with specific functionality (Johnson, 2002). OSS software development typically has a central person or body that selects a subset of developed code for an “official” release and makes it widely available for distribution. OSS is built by potentially large numbers of volunteers in combination with for-profit participants (Von Krogh, 2003). Often no system-level design or even detailed design exists. Developers work in arbitrary locations, rarely or never meet face to face, and often coordinate their activity through e-mail and bulletin boards. As participants make changes to the original application, the central person or body leading the development selects code changes, incorporates them into the application, and officially releases the next version of the application. Table 2 compares OSS to proprietary development.

OSS BUSINESS MODELS

A business model is a method whereby a firm builds and uses resources to provide a value-added proposition to potential customers (Afuah & Tucci, 2000).

Table 2. OSS development vs. proprietary development

OSS	Proprietary Software
Similarities	
Building brand name and reputation increases software use	
Revenue is generated from supporting software, creating new applications for software, and certifying software users	
Differences	
Code developed outside of company for free	Developers are paid to program code
Source code is open for public use.	Source code is kept in company
People use program without paying any license fees.	Users pay license fees to use the software
Updates are free and users are allowed flexibility in using them	People are locked in using specific software and have to pay for updates
Code is developed for little internal cost	Code is costly to create internally

OSS business models are based on providing varied services that cater to cost-sensitive market segments and provide value to the end user by keeping the total cost of ownership as low as possible (Hecker, 1999). OSS-based companies must provide value-added services that are in demand, and they must provide these services at cost-sensitive levels. OSS is a strategic threat to proprietary software, because one of the most effective ways to compete in lock-in markets is to “change the game” by expanding the set of complementary products beyond those offered by rivals (Shapiro & Varian, 1998). OSS proponents are trying to “change the game” with new applications of the following business models (Castelluccio, 2000): support sellers, loss leaders, code developers, accessorizers, certifiers, and tracking service providers.

Support Sellers

Support sellers provide OSS to customers for free, except for a nominal packaging and shipping fee, and instead charge for training and consulting services. They also maintain the distribution channel and branding of a given OSS package. They provide value by helping corporations and individuals install, use, and maintain OSS applications. An example of a support seller is RedHat, which provides reliable Linux solutions.

To offer such services, support sellers must anticipate and provide services that will meet the needs of businesses using OSS. To offer reliable and useful consulting services, support sellers must invest heavily in understanding the currently available OSS packages and developing models to predict how these OSS applications will evolve in the future (Krishnamurthy, 2003).

This model has strengths in meeting the needs for outsourcing required IT services, which is the current market trend (Lung Hui & Yan Tam, 2002). OSS provides companies an opportunity to reduce licensing costs by allowing companies to outsource the required IT support to support sellers. Likewise, the marketplace structure of OSS development adds significant uncertainty to the future of OSS applications. Risk-averse companies often do not want to invest in specialized human capital, and support sellers help mitigate these risks.

One drawback of this model is that consulting companies often fall prey to economic downturns, during which potential clients reduce outsourcing to consultants. This cycle is compounded for the software industry, since a poor economy results in cost cutting and an eventual reduction in IT spending.

Loss leaders

Loss-leader companies write and license proprietary software that can run on OSS platforms (Castelluccio, 2000). An example of a loss leader is Netscape, which gives away its basic Web-browser software but then provides proprietary software or hardware to ensure compatibility and allow expanded functionality. The loss-leader business model adds value by providing applications to companies that have partially integrated OSS with their systems (Hecker, 1999). Companies often need specific business applications that are unavailable in the OSS community, or they desire proprietary applications but wish to avoid high platform-licensing costs.

To leverage the integration of OSS with proprietary software, loss leaders need to assemble a team of highly skilled developers, create an IT infrastructure, and develop licensable applications. The major costs of this business model arise from payroll expenses for a development team, R&D costs, marketing, and, to a lesser extent, patenting and manufacturing.

This model’s strength is that it provides a solution for the lack of business applications circulating in the OSS community. The loss leader model fills the gap between simpler available OSS applications, such as word processors, and more complex applications that are unavailable in the OSS community.

A weakness of this model is the risk of disintermediation. As time passes and OSS coding continues to grow and expand, more robust and complex applications will be developed. However, the developers of these applications will have to cope with the speed and efficiency of proprietary software development.

Code Developers

The code development model addresses some of the limitations of the loss-leader model. Code development companies generate service revenue through

on-demand development of OSS. If a firm cannot find an OSS package that meets its needs for an inventory management system, for example, the firm could contract with a code development company to the basic application (Johnson, 2002). The code development company could then distribute this application to the OSS community and act as the development project's leader. The code development company would track the changes made to the basic source code by the OSS community and integrate those changes into its product. The company would periodically send its customers product updates based on changes accepted from the OSS community.

The necessary assets and associated costs required by this model are similar to those in the proprietary software model, including a team of programmers, IT infrastructure, and marketing. However, the code developer needs to develop only a basic application. Once the basic software is developed, the OSS community provides further add-ons and new features (Johnson, 2002), which decrease the R&D costs for the company acting as project leader. Yet the code development team needs to have the necessary IT infrastructure to lead the OSS community in the application's evolution, incorporate new code, and resubmit new versions to its customers.

This model's strength is its longevity. The code development model overcomes the risk of disintermediation by basing its revenue generation on initiating OSS applications and maintaining leadership over their evolution; it does not focus on privatizing the development and licensing of applications.

This model's weakness is the risk of creating an application of limited interest to the OSS community. A possible solution to this problem would be an offer from the company leading the development process to reward freelance developers for exceptional additions to the application's original code.

Accessorizers

Accessorizers companies add value by selling products related to OSS. Accessorizers provide a variety of different value-added services, from installing Linux OS on their clients' hardware to writing manuals and tutorials (Hecker, 1999; Krishnamurthy, 2003). For example, O'Reilly & Associates, Inc. writes manuals for OSS and produces downloadable copies of Perl, a programming language.

One strength of this model is that it provides the new manuals and tutorials that the constantly changing nature of the OSS market requires. Another strength is its self-perpetuating nature: as more manuals and tutorials are produced, more people will write and use OSS applications, increasing the need for more manuals and tutorials.

This models' weakness is the difficulty of staying current with the many trends with the OSS community. This difficulty creates the risk of investing in the wrong products or producing too much inventory that is quickly outdated.

Certifiers

Certifiers establish methods to train and certify students or professionals in an application. Certificate companies like CompTIA generate revenue through training programs, course materials, examination fees, and certification fees. These programs provide value to the individuals enrolled in the certification programs and businesses looking for specific skills (Krishnamurthy, 2003). Certification helps the OSS industry by creating benchmarks, expectations, and standards employers can use to evaluate and hire employees based on specific skill sets.

Certification has long-term profit potential since most certification programs require recertification every few years due to continuing education requirements. Businesses value certification programs because they are a cost-effective way to train employees on new technologies. Certifiers, who achieve first-mover advantage, become trendsetters for the entire industry, increasing barriers to entry into the certification arena.

One downside of this model is the significant startup costs. Certifiers need to find qualified individuals to create manuals, teach seminars, and write tests. Certifiers must also survey businesses to discern which parts of specific applications are most important, and which areas need the greatest focus during training. Certifiers also need to gain substantial credibility through marketing and critical mass or their tests have little value. Increasing company name recognition and building a reputation in the certification arena can be an expensive and long process.

This model also faces the threat of disintermediation. Historically, certification programs have evolved into not-for-profit organizations, such

as the AICPA in accounting, or the ISO 9000 certification in operations. The threat of obsolescence is another major weakness. In the 1970s, FORTRAN or COBOL certification may have been important (Castelluccio, 2000), but they have since become obsolete. Certifiers specializing in certain applications must be constantly aware of the OSS innovation frontier and adjust their certification options appropriately.

Tracking Service Providers

The tracking-services business model generates revenue through the sale of services dedicated to tracking and updating OSS applications. For example, many companies have embraced Linux to cut costs; however, many of these same companies have found it difficult to maintain and upgrade Linux because of their lack of knowledge and resources. Tracking-services companies, like Sourceforge.net and FreshMeat.net, sell services to track recent additions, define source code alternatives, and facilitate easy transition of code to their customers’ systems.

A strength of this model is its ability to keep costs low by automating the majority of the work involved in tracking while still charging substantial subscription and download fees. However, these services must have Web-based interfaces with user-friendly download options, and they also must develop human and technological capabilities that find recent updates and distinguish between available alternatives.

A weakness of this model is low barriers to entry. This information-services model can be replicated with a simple Web interface and by spending time on OSS discussion boards and postings, creating the possibility of such services becoming commoditized. Table 3 summarizes some of the differences between the OSS business models.

CONCLUSION

The market battle between OSS and proprietary software has just begun. This battle could be termed a battle of complementary goods and pricing. For example, the strategies between Microsoft and RedHat are similar in that they both need a large, established user base that is locked in and has access to a large array of complementary goods and services. The key differences in their strategies are in their software development process, software distribution, intellectual property ownership, and pricing of core products and software. It will be increasingly important for OSS companies to track the competitive response of proprietary companies in combating the increasing presence of OSS.

Moreover, the OSS movement has begun to make inroads into the governments in China, Brazil, Australia, India, and Europe. As whole governments adopt OSS the balance of power can shift away from proprietary providers. This also provides the opportunity to develop a sustainable business model that caters only to the government sector. Similarly, for-

Table 3. OSS models

Business Model	Assets	Costs	Revenue Model
Support Sellers	Human capital, supporting infrastructure, contracts	Payroll, IT, marketing and brand development	Training, consulting
Loss Leaders	Human capital, supporting infrastructure, software	Payroll, IT cost, marketing and brand development, R&D, software manufacturing	Licenses
Accessorizers	Human capital, supporting infrastructure	Payroll, printing material machines, training, software	Book Sales
Code developers	Human capital, software - technology tracking, database	Payroll, IT, marketing (Corporations), marketing (Freelancers)	Corporations that pay for service
Certifiers	Human capital, IT, Certification program	Certification program development, payroll	Tests, certificates
Tracking-service providers	Human capital, Software-technology tracking, Databases	Payroll, IT, marketing (Corporations)	Corporations that pay for service

mulating business models for corporations and educational institutions may be another fruitful opportunity.

The recent government regulations associated with the Sarbanes-Oxley Act and other financial-reporting legislation are important trends. These regulations require significant research in the area of internal control reporting on OSS applications. It is likely the collaborative and less proprietary nature of OSS could help with this reporting. If this reporting can be done with more assurance than provided by proprietary applications, OSS providers can gain further advantage.

REFERENCES

Afuah, A. & Tucci, C. (2000). *Internet business models and strategies: Text and cases*. McGraw-Hill Higher Education.

Bretthauer, D. (2002). Open source software: A history. *Information Technology & Libraries*, 21(1), 3-10.

Castelluccio, M. (2000). Can the enterprise run on free software? *Strategic Finance*, 81(9), 50-55.

Christensen, C.M. (1997). *The innovator's dilemma: When new technologies cause great firms to fail*. Harvard Business School Press.

DeLong, J.B. & Froomkin, A.M. (2000). Beating Microsoft at its own game. *Harvard Business Review*, 78(1), 159-164.

Fitzgerald, B. & Feller, J. (2001). Guest editorial on open source software: Investigating the software engineering, psychosocial and economic issues. *Information Systems Journal*, 11(4), 273-276.

Hecker, F. (1999). Setting up shop: The business of open-source software. *IEEE Software*, 16(1), 45-51.

Johnson, J.P. (2002). Open source software: Private provision of a public good. *Journal of Economics & Management Strategy*, 11(4), 637-662.

Krishnamurthy, S. (2003). A managerial overview of open source software. *Business Horizons*, 46(5), 47-56.

Lederer, A.L. & Prasad, J. (1993). Information systems software cost estimating: A current assessment. *Journal of Information Technology*, 8(1), 22-33.

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

Lung Hui, K. & Yan Tam, K. (2002). Software functionality: A game theoretic analysis. *Journal of Management Information Systems (JMIS)*, 19(1), 151-184.

MacCormack, A. (2001). Product-development practices that work: How Internet companies build software. *MIT Sloan Management Review*, 42(2), 75-84.

Shapiro, C. & Varian, H.R. (1998). *Information rules: A strategic guide to the network economy*. Harvard Business School Press.

Von Krogh, G. (2003). Open-source software development. *MIT Sloan Management Review*, 44(3), 14-18.

KEY TERMS

Copyright: A legal term describing rights given to creators for their literary and artistic works. See World Intellectual Property Organization at www.wipo.int/about-ip/en/copyright.html.

General Public License (GPL): License designed so that people can freely (or for a charge) distribute copies of free software, receive the source code, change the source code, and use portions of the source code to create new free programs.

GNU: GNU is a recursive acronym for "GNU's Not Unix." The GNU Project was launched in 1984 to develop a free Unix-like operating system. See www.gnu.org/.

Open-source Software (OSS): Software that can be freely used in the public domain, but is often copyrighted by the original authors under an open-source license such as the GNU GPL. See the Open Source Initiative at www.opensource.org/docs/definition_plain.php.