An Analysis of Open Source Business Models

Sandeep Krishnamurthy
Associate Professor of E-Commerce and Marketing
Business Administration Program
University of Washington, Bothell
18115 Campus Way NE, Room UW1-233
Bothell, WA 98011-8246

Tel: (425) 352-5229
Fax: (425) 352-5277
E-Mail: sandeep@u.washington.edu
Web URL: http://faculty.washington.edu/sandeep

February 2003

Introduction

Open-source software products provide access to the source code [or basic instructions] in addition to executable programs, and allow for this source code to be modified and redistributed. This is a rarity in an industry where software makers zealously guard the source code as intellectual property.

In making the source code freely available, a large number of developers are able to work on the product. The result is a community of developers spread around the world working to better a product. This approach has led to the popular operating system, LINUX, which has emerged as a credible threat to Microsoft's products- especially on the server side. Other famous open-source products include Apache [a program used to run websites], OpenOffice [an alternative to Microsoft Office] and Sendmail [the program that facilitates the delivery of approximately 80% of the world's e-mail].

Open-source is typically viewed as a cooperative approach to product development and hence, more of a technology model. It is typically not viewed as a business approach. However, increasingly we find that entire companies are being formed around the open source concept. In a short period of time, these companies have amassed considerable revenues [although it is fair to say that most of these firms are not yet profitable].

Consider two companies in particular- Red Hat and Caledera/SCO. In its last full year of operations [12 months ending February 28, 2002], Red Hat's revenues were almost $79 million. In its last full year of operations [12 months ending October 31, 2002] Caldera/SCO's revenues were about $64 million. The growth figures are even more impressive- Caldera/SCO grew its revenue from $1 million in 1998 to $64 million in 2002 and Red Hat grew from $42 million in 2000 to $79 million in 2002.

All software companies exist to make maximum profits. Therefore, it is common for these corporations to seek out new ways of generating revenues and reducing costs. Increasingly, companies are using open-source as a business strategy to achieve both these objectives.

On the cost reduction side, software producers are now able to incorporate the source code from an open-source product into an existing code base. This allows them to reduce the cost of production by reusing existing code. For example, Microsoft, the world's largest software maker, has used source code from a leading open-source operating system [Berkeley System Distribution or BSD] in its Windows 2000 and XP products and has acknowledged this on a public web site[i]. It is becoming more common for companies to forge strategic alliances with communities of open-source software developers. The community develops the product and thus, reduces the cost burden on the company. A prime example of this is the strategic alliance between Ximian and Microsoft in building a connection between the .Net initiative and LINUX.

On the revenue side, some open-source products are now in such great demand that there is a strong need for support services for enterprise customers. These support services includes installation, training/certification and ongoing technical assistance.

Service contracts for these products have become a strong revenue source for companies such as Red Hat Linux.

From the consumer perspective, open source products are attractive due to their reduced cost and comparable performance. Governments, for example, are increasingly motivated to adopt open-source products to reduce the expenditure of scarce taxpayer money. Some governments [e.g. Argentina] have experimented with moving entirely to an open source model.

Even for individual consumers, open source products are becoming accessible. Wal-Mart has started to carry PCs that run LINUX. Many free applications are now available for PCs. For example, OpenOffice and Koffice are free, open-source products that directly compete with Microsoft's famous Office Suite.

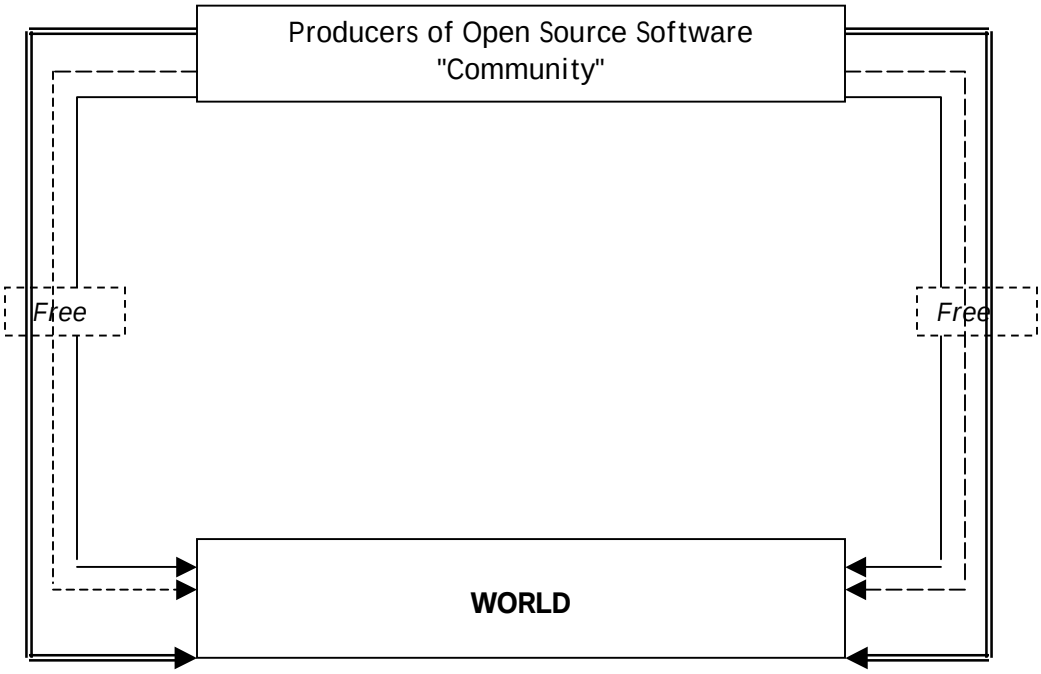
In this Chapter, my focus is on explicating the different business models that we see in the Open Source arena.

Producers of Open-Source Products - The Community

The producers of open-source products are typically a diverse group of developers with a shared passion for a product. They do not seek a profit and no distinction is made between corporate and individual users.

Therefore, they make (a) the product and (b) the source code available for free to any interested user. There is usually support available through electronic mailing lists and USENET groups. Members participate to learn more about the product and believe that others will help them if they have a need [Lakhani and Von Hippel 2003]. Surprisingly, the customer support provided by communities surrounding products such as Apache and Linux have won awards for excellence.

Figure 1
Producers of Open-Source Products



Legend: ——— Product
– – – – Source Code
═══ Service

The community of producers is frequently portrayed as being inimical to corporate profits. However, I will submit that the community is *simply indifferent* to its own profits as well as profits that any corporation can make from its products. Open source developer communities are frequently interested in adoption of the product by the intended target audience. Importantly, they want any interested developer to have access to the entire code so that the person can tinker with it to make improvements.

The community does not distinguish between a corporate or individual user. There is no sense of direct competition with companies. A company that views a community as its competitor is welcome to look at its entire source code whereas the opposite is never true. Communities do not distinguish between users across countries. When the product is available for free, it is amazingly easy to make a product global. There is no issue of taxation or piracy.

The community controls what happens with the product by making one crucial choice - the license. The original developers control the copyright for the intellectual property at all times. However, there is considerable variation between licenses in how derived works may be distributed.

There are a number of licenses that communities can choose from. However, they can be broadly classified as the GNU General Public License [GPL] and everything else. The GPL is the most famous license and products such as LINUX are distributed using it. The key feature of the GPL is that it restricts the terms of distribution of derived works. If a company incorporates GPLed source code in its products, it must make the source code for any product it sells in the marketplace available to any interested party under the terms of the GPL. This frightens corporations interested in selling open-source products. However, it is important to note that there are a whole host of other licenses that do not have this stipulation.

In my view, the derived works clause is so powerful that it affects how business models are constructed. The discussion about business models is therefore broken down into the GPL and the non-GPL model. Generally speaking, the use of GPL reduces the profit potential of companies.

It is very important to note that the community does not set a price on a software product. Even in the case when the product is available for free, anybody can incorporate the product and sell it for a price. Even with a GPL license, this is possible. Obviously, in the case of GPL, there is the attendant duty of making the source code for derived works freely available.

Business Models

In this section, I will discuss the main business models built around the open source philosophy. My focus in this section is mainly on the software/service side. It is certainly true that some companies will benefit from the sale of hardware that runs open source products. Similarly, the market for embedded products can be great. However, for the purposes of this chapter, I will focus on the software and service-oriented business.

The Distributor

The distributor provides access to the source code and the software. In the case of LINUX, leading distributors include Red Hat, Caldera and SUSE. Distributors make money in these ways-

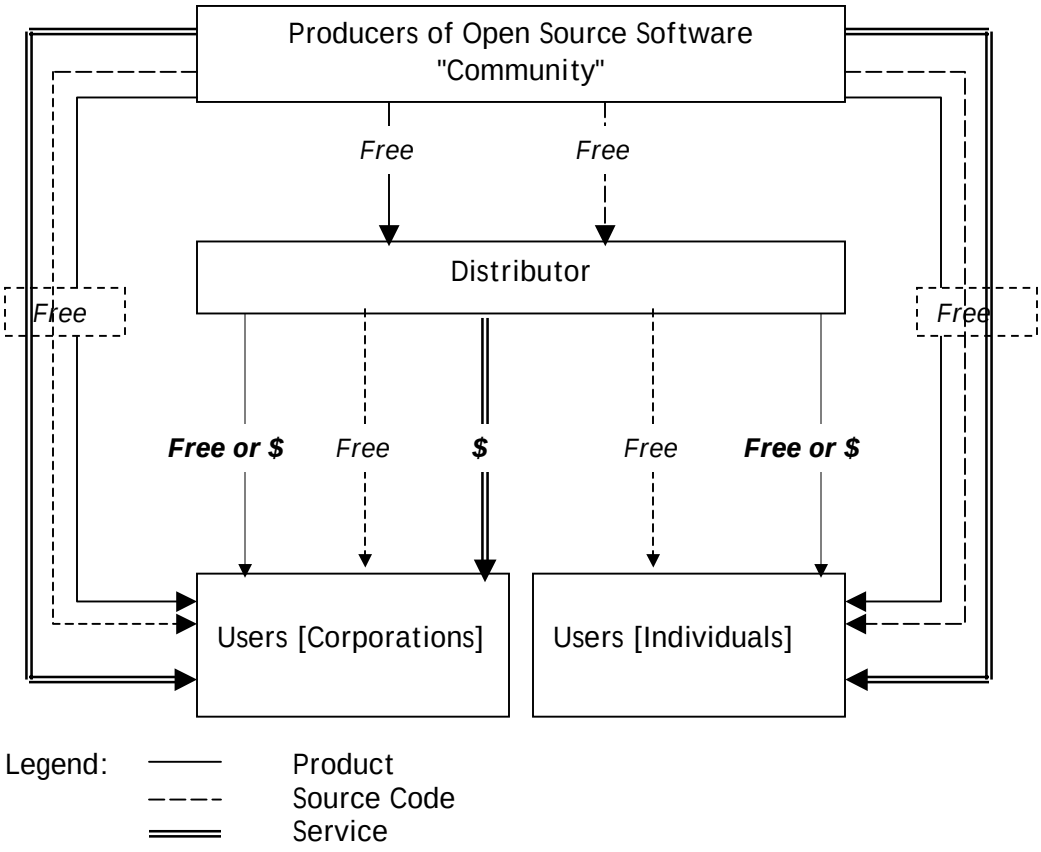
[1] Providing the product on CD rather as an online download - most people are not comfortable with downloading the product from a web site. One survey of 113,794 Linux users indicated that 37.06% of respondents preferred to obtain LINUX in CD form[ii]. Therefore, there is money to be made selling the product in CD form. According to one source [www.distrowatch.com], as of Feb 2003, the highest price that was being charged for a Linux CD was $129 [Lindows] and the lowest price for a CD was zero [e.g. Debian, Gentoo].

[2] Providing support services to enterprise customers - Enterprises are willing to pay for accountability. When they have a problem, they do not want to send a message to a mailing list and wait for support that may or may not be of the highest quality. They have no interest in sifting through technical FAQs to find the answer. Therefore, there is money to be made in services such as support for installation, answering technical questions and training employees to use the product.

[3] Upgrade Services - Enterprises can now enter into long-term agreements with distributors to ensure that they get the latest upgrade. By acting as application service providers, distributors can help their clients get the latest version of the product seamlessly.

The business model of distributors is shown in Figure 2 below.

Figure 2
The Distributor Business Model



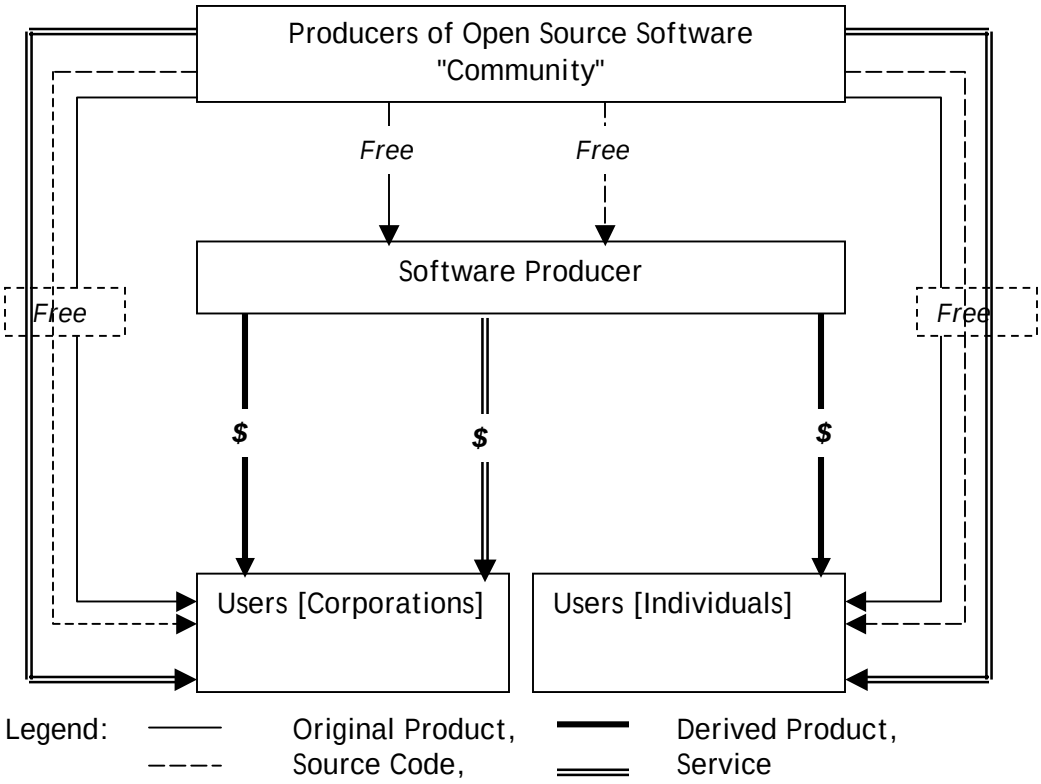
The Software Producer [Non-GPL Model]

Software producers can benefit from the open source software community in two ways. First, they can incorporate the source code of an existing product in a larger code base and create a new product. Second, they can also take an entire open source product and bundle it with existing products. I am using the term "derived product" in a very general sense here to include both these cases. The source code for the derived product does not need to be disclosed since the license is not GPL.

As mentioned earlier, Microsoft has incorporated the code from BSD in its products and has not released the source code to any interested party. All Microsoft had to do was to acknowledge that it benefited from BSD's code.

The software producer benefits from lowered cost of production and hence, increased margin in this case. There is a service revenue stream in place here as well. The business model itself is shown in Figure 3.

Figure 3
Software Producer-Non-GPL Model

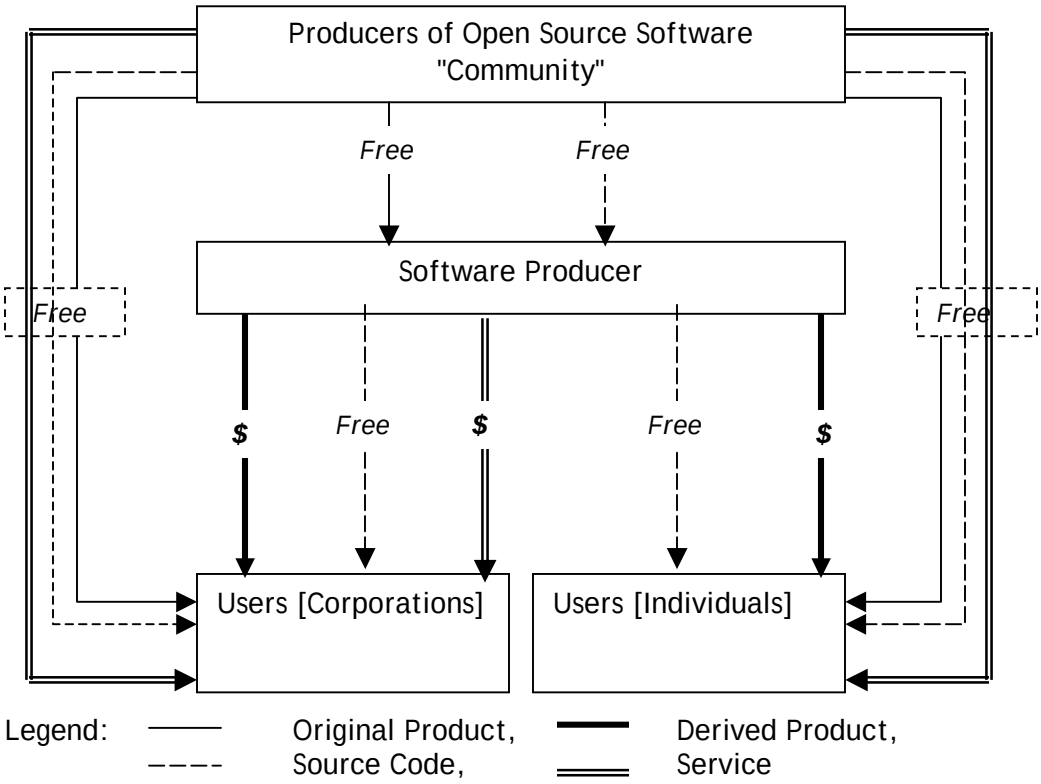


Interestingly, the source code for the original product is still available to the end users from the community. In the cases where the derived product is a small adaptation of the original product, this may be very useful to the end users. This is the cost the for-profit software producer pays to get the source code for free.

Software Producer- GPL Model

The business model for this case is shown in Figure 4.

Figure 4
Software Producer - GPL Model



The key difference between Figures 3 and 4 is that in the latter, the software producer is forced to make the source code for the derived product available to the end user.

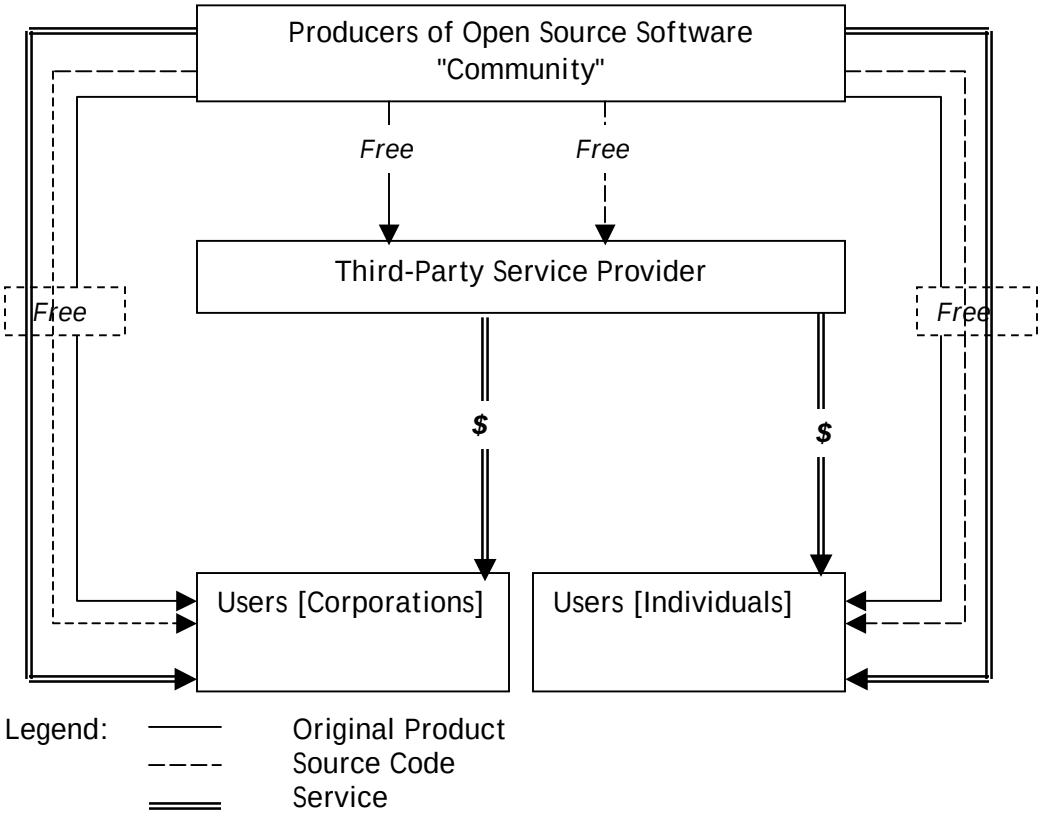
Let us compare the GPL and non-GPL models. The release of the source code in the GPL model accelerates innovation due to more rapid feedback and input. Greater inclusion of users builds relationships and hence, loyalty. Also, if the user builds a new version of the product for commercial use, the company gets to see it along with the source code. However, it does expose the inner workings of the company's product to the users.

Ultimately, the difference between the GPL and non-GPL models is in terms of what the seller expects from the user. The GPL software producer expects an empowered user who is eager to engage in a two-way conversation. The non-GPL software producer wants the recipient of the software to simply use it and do nothing else.

Third Party Service Provider

The mission of third party service providers is simple. They don't care where you got the code or where you got the product. If the product you are using meets a broad set of criteria, they will fully support it. They have one single revenue stream- service. Their business model is shown in Figure 5.

Figure 5
Third Party Service Provider



Why should users - especially corporations - use these providers? The bottom line is that paid service generally equates to higher quality service. Moreover, in many cases, third party service providers are local and hence, may be able to provide on-site assistance that is typically not possible in the case of free service on mailing lists and user groups. It is important to keep in mind that these service providers are competing with the community to provide customer service.

I have presented two types of models here- one where the company sells software and service and one in which a company simply offers a service. It is interesting to speculate if a company can survive on the sale of software alone.

Surviving on the sale of software alone is not easy to achieve. Remember that the community is already making a free version of the product available. The company must be able to add considerable value to the product to generate sufficient margins.

How can a company add value? First, it can choose a version of the product that is stable and that is most suited to its users' needs. Second, it can create a suite of products that are well integrated. These products may come from different sources- some open-source, some commercial. The value addition is in creating one package that works well together.

In general, we find that sale of software alone is insufficient to sustain. What is needed is software and service. For many software sellers, they already have a relationship with enterprise customers. They can benefit most by up-selling- i.e., selling more to existing corporate customers. Selling service then becomes a logical conclusion.

Advantages and disadvantages of Open-Source

Let us now take a close look at the potential advantages and disadvantages of using open-source technology to develop new products.

Advantages

1. Robustness

   Traditionally, a company hires a finite number of developers to craft the software. Next, a group of testers work with the product to make sure the number of bugs is minimized. At that point, it is launched to the market. In direct contrast, with the open-source method, a much larger number of developers and testers can work on the product and test it under a variety of conditions.

   The open-source method could potentially lead to a more robust product. The term robust here refers Neumann's sense- i.e., an intentionally inclusive term embracing meaningful security, reliability, availability, and system survivability, in the face of a wide and realistic range of potential adversities (Neumann 1999). Open source leaders have long maintained that this methodology leads to greater reliability (Ghosh 1998).

   Several studies corroborate this. A study by Bloor Research clearly demonstrated the superiority of Linux over Windows NT (Godden 2000). A study conducted by Netcraft in August 2001 found that 92% of the top 50 often-requested sites with the longest uptimes ran Apache [uptime.netcraft.com].

2. Flexibility to user

   One of the problems with regular software programs is that unless you work with all the software from one company, you do not have the flexibility of "mixing and matching". In the words of Linus Torvalds(Ghosh 1998),

   > In fact, one of the whole ideas with free software is not so much the price thing and not having to pay cash for it, but the fact that with free software you aren't tied to any one commercial vendor. You might use some commercial software on top of Linux, but you aren't forced to do that or even to run the standard Linux

kernel at all if you don't want to. You can mix the different software you have to suit yourself.

3. Support from a community

   Traditionally, if a user has a problem, he or she has to contact the technical support division of the company. In many cases, the level of support is poor (especially in the case of free service) or the user may have to pay a fee to get high-quality service. Moreover, after a point, users are asked to pay for this support. With open-source software, one has a highly motivated community willing to answer questions (Lakhani and Von Hippel 2003). In the case of Linux, Linux User Groups [or LUGs] are numerous and do an excellent job providing service.

Disadvantages

Even though open-source product development has a lot of positives, it also comes with its share of negatives.

1. Version Proliferation.

   Consider the data in Table 2. This is based on the survey of 3568 machines. The count is the number of machines and the % is the percentage of machines running a particular version. As shown in the Table, there are at least 62 versions of the software running at this time.

   The reason for this multiplicity of versions is due to a complicated version release structure employed by LINUX. Releases can either by even-numbered or odd-numbered. The former represent relatively stable software that can be used by enterprise customers. In particular, version 2.0 and 2.2 were major releases that were a long time in the making. On the other hand, odd-numbered releases are developmental versions of the product with new product features. This complicated structure was employed to satisfy two audiences- developers and enterprise customers [Sproull and Moon 2000].

   This makes it very difficult for the end-user to identify the best version of the product. Companies such as Red Hat, play an important role here by selecting one version to support.

Table 2, Survey of LINUX Kernel Versions

(Source: Alvestrand, Harald, "The Linux Counter Project", <www.linuxcounter.org>, Accessed on February 12, 2002)

Number	Kernel	Count	Percentage	Number	Kernel	Count	Percentage
1	2.0.28	3	0.10%	58	2	33	0.90%
2	2.0.32	2	0.10%	59	2.2	488	13.70%
3	2.0.33	2	0.10%	60	2.4	3019	84.60%
4	2.0.34	2	0.10%	61	2.5	25	0.70%
5	2.0.34C52_SK	2	0.10%	62	Others		0.10%
6	2.0.36	6	0.20%				
7	2.0.37	4	0.10%				
8	2.0.38	5	0.10%				

9	2.0.39	3	0.10%		
10	2.2.10	2	0.10%		
11	2.2.12	10	0.30%		
12	2.2.13	15	0.40%		
13	2.2.14	34	1.00%		
14	2.2.15	2	0.10%		
15	2.2.16	62	1.70%		
16	2.2.17	23	0.60%		
17	2.2.18	23	0.60%		
18	2.2.18pre21	4	0.10%		
19	2.2.19	126	3.50%		
20	2.2.19ext3	5	0.10%		
21	2.2.19pre17	11	0.30%		
22	2.2.20	69	1.90%		
23	2.2.20RAID	2	0.10%		
24	2.2.21	11	0.30%		
25	2.2.22	29	0.80%		
26	2.2.23	10	0.30%		
27	2.2.24	8	0.20%		
28	2.2.25	24	0.70%		
29	2.2.5	9	0.30%		
30	2.4.0	6	0.20%		
31	2.4.10	42	1.20%		
32	2.4.12	10	0.30%		
33	2.4.13	9	0.30%		
34	2.4.14	12	0.30%		
35	2.4.16	48	1.30%		
36	2.4.17	63	1.80%		
37	2.4.18	1056	29.60%		
38	2.4.19	391	11.00%		
39	2.4.2	44	1.20%		
40	2.4.20	942	26.40%		
41	2.4.20.1	2	0.10%		
42	2.4.21	178	5.00%		
43	2.4.3	13	0.40%		
44	2.4.4	28	0.80%		
45	2.4.5	9	0.30%		
46	2.4.6	7	0.20%		
47	2.4.7	54	1.50%		
48	2.4.8	18	0.50%		
49	2.4.9	46	1.30%		
50	2.4.x	2	0.10%		
51	2.5.63	2	0.10%		
52	2.5.65	2	0.10%		
53	2.5.66	2	0.10%		
54	2.5.67	4	0.10%		
55	2.5.68	4	0.10%		
56	2.5.69	6	0.20%		
57	Others		1.70%		

2. Usability

> Some open source products suffer from poor usability (Nichols and Twidale 2003). This may stem from the way projects are structured, the nature of the audience and the level of resources available to open source projects. However, for major products [i.e., Stars], this is an opportunity for a new business.

Analyzing the Profit Potential of Open Source Products

Not all open source products have a high profit potential. To analyze the profit potential of an open source product, I use two dimensions - customer applicability and relative product importance. The classification scheme that results from this is shown in Figure 7.

Customer applicability refers to the proportion of the market that can benefit from the software. For example, if a product is being designed for a rarely used operating system, only a small proportion of consumers will be able to benefit from it. This will make the level of customer applicability small. On the other extreme, some products are designed for a large number of computing environments or the computing environment that is most commonly found. This makes it high on customer applicability.

Relative product importance refers to how important a program is to the functioning of the user's computer. An operating system is clearly the most important. Without it, the computer will not be able to function. On the other extreme, a screensaver program will add some value to the user- but it is something that the user can do without.

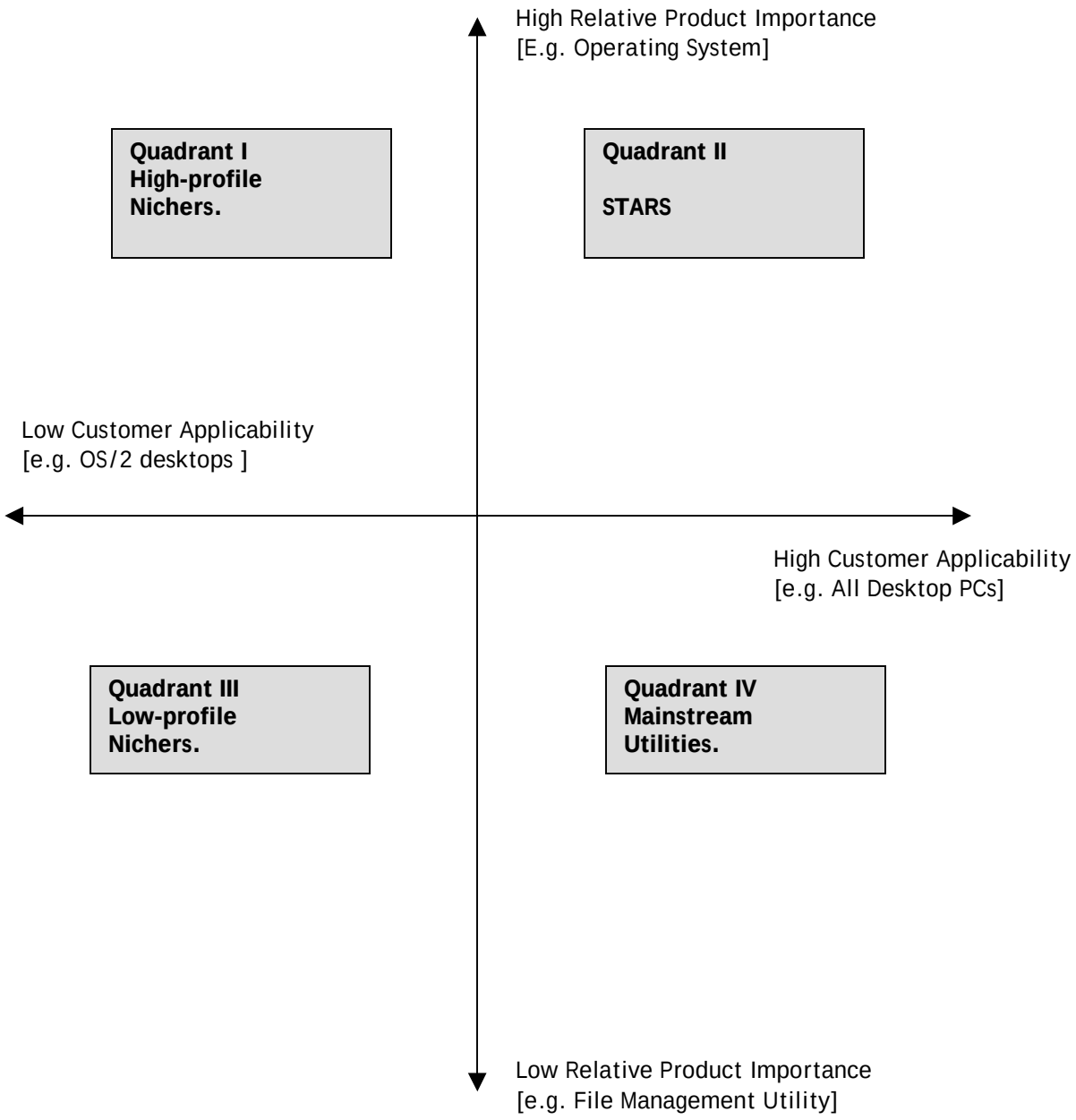
The products with the highest profit potential have high relative product importance and high customer applicability [Quadrant II in Figure 7]. These are the stars that we hear most about. Companies are started around these products. They have large developer communities supporting them. These products have the greatest direct and indirect marketing support. These products have the highest profit potential. An example of such a product is LINUX. Its relative importance is high since it is an operating system and its customer applicability is high since it can be installed on every desktop PC.

On the other extreme, products that are low relative product importance and low customer applicability are the low-profile nichers [Quadrant III in Figure 7]. These products serve a specific niche and itch a small scratch (Raymond 1998). They are never going to be dominant products that will run on a large proportion of desktops. But, that is not the goal of the creators of these products. The creators know they are filling a small niche and their goal is to fill it effectively. These products have the lowest profit potential. A good example of such a product is Wings3D, which is a very powerful polygon mesh modeler. This is perhaps a program that students of advanced mathematics may find useful.

The products with low relative product importance and high customer applicability are the mainstream utilities [Quadrant IV in Figure 7]. These are products that everybody can benefit from. However, they are not critical to the functionality of the computer. For instance, TouchGraph's Google Browser converts the search results within result into a graphical map. This makes for an interesting map of the results. However, it may not be something, by itself, that is commercially feasible. Another great example of a mainstream utility is Agnostos- a Web-based tool for managing to-do lists. Such products could make excellent promotional items for companies.

Finally, the products with high relative product importance and low customer applicability are the high-profile nichers [Quadrant I in Figure 7]. These products are regarded very highly within the specific niche that they serve. However, beyond that, they are not well known. If marketed well, they can lead to a profitable operation. A great example of this is SquirrelMail. This is a program that can be used to run an Internet Service Provider's (ISP) mail operation. It is very well regarded within its niche.

Figure 7
Classification of Open-Source Products



Why Should Corporate Users Switch to Open-Source Products?

There are three ways to respond to this question.

The first issue is product performance. Large companies will not adopt a product just because it is built using a certain product development style. They care about performance. Open-source products have been making inroads into large companies because they are good- it is just that simple. In many cases, open-source products have been evaluated for their technical merits and their ability to meet stringent requirements. They have been adopted because they met and exceeded these requirements. Examples of notable adoptions include Amazon and Yahoo's use of Perl, Orbitz' use of Linux and Apache and Google's usage of Linux.

Second, since open-source products are usually available for free as an online download, corporations can treat it as a low product risk. They can simply download the product and play with it in a back-office for a while. Even if they decide not to implement it, they will have not paid anything. Of course, this only covers the up-front cost of purchasing the product (see next point about total cost of ownership).

Third, corporations must evaluate the total cost of ownership [i.e., the cost of purchasing, installing and maintaining the product] of corporate alternatives with open-source products and see what that tells them. If the total cost of ownership is in fact lower with open source products, there may be a case. The total cost of ownership is sensitive to the nature of the organization and should be evaluated by each organization as such.

Key Factors That Affect Profits

Support from primary developer community

The key engine for innovation within the open source ecosystem is the primary developer community (Shankland 2002). If this community is focused on innovation, everybody benefits. Distributors can use the latest version in their next release. Software producers can add the latest code. Customers get the product with the best performance that is most stable.

The success of a developer community crucially depends on its leadership structure. However, a variety of leadership styles and structures are observed. For instance, Linus Torvalds is generally considered to be a strong leader in all senses of the word. On the other hand, a committee runs Apache. At this time, it seems like the issue is clarity of the direction for the project. This may be provided by one leader or a group of people working closely together.

Presence of dominant competitive OSS products

OSS products compete with each other fiercely. Open source products compete for developers, distributors and customers. Developers want to be associated with products that are likely to have a major impact. Distributors would like to devote

resources only to products that are likely to become very successful. Customers want to use products that they can rely on.

There are two levels of competition- the product category level [E.g. BSD and LINUX are competing open source operating systems] and the distribution level- the distributors of LINUX are in aggressive competition with each other.

The competition among LINUX distributors is especially interesting. Red Hat has established a dominant position- especially in the American market. One source puts its market share in the 50% range[iii]. However, many other distributors are vying for share. Recently, four LINUX distributors- Caldera, Conectiva, SuSE, and TurboLinux- have decided that instead of competing with one another, they must compete with the market leader, i.e., Red Hat. To this end, they have formed a group called UnitedLinux. This company will release one product that all four will support. However, each individual company retains its identity and will strive to differentiate on the service side.

While some competition may be necessary for product innovation, excessive competition can hamper long-term profitability.

Presence of dominant competitive non-open-source products

Perhaps, the greatest threat to profits from an OSS product is the presence of competitive non-OSS products. Linux competes with Microsoft's Windows products. OpenOffice competes with Microsoft Office. Products such as OpenCourse and Moodle compete with commercial products such as WebCT and Blackboard in the course design arena.

In all these cases, the commercial competitor has a resource advantage that can be used to gain market power through advertising, salesperson interaction with large corporations and through public relations. Sometimes, the presence of such competition creates an underdog mentality that can help the open-source product to some degree. On the other hand, it is very hard to compete with major corporations on a regular basis.

Relative competitive position

In the final analysis, what really matters is how competitive the product is. If the product is truly innovative, it will have a strong chance. If it is does not stack up well against competitive products, it will not. The hope is that making the source code available for free will lead to greater innovation. However, this may fail to materialize if a software product does not attract too many developers.

Need for marketing

Building awareness for open source products is a challenge. Consider the case of LINUX. There is a two-level challenge here. On the first level, one must build awareness for LINUX itself [product category awareness]. On the second level, one must create awareness for a specific distribution- such as Red Hat [brand awareness]. Distributors will only be interested in boosting brand awareness. Red Hat will want to

be closely associated with LINUX and they would want people to equate LINUX with their brand name.

If there are no companies in the market, the community will have to take on this challenge. In that case, awareness is built using techniques such as word of mouth that are not resource-intensive.

Of course, building awareness alone is insufficient. What is needed is greater product knowledge followed by trial of the product.

Conclusion

We now know that it is possible to build a business around the open-source strategy. We are increasingly finding that Open Source Software communities are awesome competitors. They are able to compete with large companies on an equal footing and even defeat them. They are, therefore, not to taken lightly or dismissed off-hand.

Open-source software is not for hobbyists any more. Instead, it is a business strategy with broad applicability. Businesses can be built around this idea. In this paper, I want the reader to grapple with the specifics of how to build and grow such a business.

To this end, I have proposed three fundamental business models- Distributor, Software producer [GPL and non-GPL] and the Third-Party Service Provider. These are sustainable models that can lead to robust revenue streams. The business models provided here can be enhanced by the addition of further revenue streams. For instance, we now know that certification of developers on an Open-Source product can lead to strong revenues.

Not all products have the same profit potential. Therefore, not all Open Source Software products have the same profit potential. I have classified Open Source Software products into four categories- Stars, High-profile nichers, Low-profile nichers and Mainstream utilities. Businesses can be built around Stars. High-profile nichers can lead to robust revenue streams if properly marketed. The other two categories may not lead to high profits. Since many Open Source Software products are freely available, managers must scan public repositories to find out which products will be suitable for their business.

The future of Open Source Software is bright. Increasingly, we will find that these products will take a central role in the realm of software and will find a larger place in all our lives.

REFERENCES

Ghosh, Rishabh Aiyer (1998), “What Motivates Free Software Developers: Interview with Linus Torvalds”, 3(3), *First Monday*, Available at <http://www.firstmonday.dk/issues/issue3_3/torvalds/index.html>.

Godden, Frans (2000), " How do Linux and Windows NT measure up in real life?", Available at- <http://gnet.dhs.org/stories/bloor.php3>.

Leonard, Andrew (1998), “Let My Software Go”, Salon, Available at- <http://www.salon.com/21st/feature/1998/04/cov_14feature2.html>.

Lakhani, Karim and Eric Von Hippel (2003), "How Open Source Software Works: Free User-to-User Assistance", Forthcoming in *Research Policy*.

Moon, Jae Yun and Lee Sproull (2000), "Essence of Distributed Work: The Case of the Linux Kernel", *First Monday*, 5(11), Available at- http://firstmonday.org/issues/issue5_11/moon/index.html

Neumann, Peter G. (1999), "Robust open-source software", *Communications of the ACM,* 42(2), 128-129.

Nichols, David M. and Michael B. Twidale (2003), "The Usability of Open Source Software", *First Monday*, 8(1), Available at: <http://firstmonday.org/issues/issue8_1/nichols/index.html>.

Raymond, Eric (1998), "The Cathedral and the Bazaar", 3(3), Available at- <http://www.firstmonday.org/issues/issue3_3/raymond/index.html>.

Shankland, Stephen (2002), "Tiemann steers course for open source", ZDNet, December 4, Available at- <http://zdnet.com.com/2100-1104-975996.html>.

ENDNOTES

[i] See- http://support.microsoft.com/default.aspx?scid=KB;en-us;q306819
[ii] http://counter.li.org/reports/machines.html, Accessed on Feb 09, 2002.
[iii] <http://www.newsfactor.com/perl/story/20036.html>