

## **The Business and Economics of Linux and Open Source**

December 2006

Martin Fink is responsible for driving Hewlett-Packard's Linux and open source strategy. He heads HP's Business Critical Systems which includes not only the open source and Linux operations but also NonStop servers and business critical servers. His book<sup>1</sup> is a clear and concise explanation of the advantages and pitfalls of open source for those management teams who are reluctant to move to open source because they don't understand the business model. It is also valuable for those who have breached that barrier and are anxious to learn more. As such, this book is for management, not for developers (nary a code statement in the entire book).

The following review is an encapsulation of Martin's easy-to-read and timely book.

### **Introduction**

The concept of open source software can be incredulous to many. How can I trust my business to it? What control mechanisms drive it? How do I support it if I am not buying it from some vendor? Is it really free; and if so, how do people make money on it? After all, if there is no money to be made, this movement isn't going to last very long.

First of all, open source software is known as free software. But the first lesson to learn is that "free" doesn't mean free of cost. It means the freedom to use, modify, and distribute the software. The existence of a large community of talented people continually improving a software product is the basis for the strength of open source software. The only requirement is to comply with the open source licensing agreement.

Perhaps the leading open source project is the Linux operating system kernel. It is estimated that 40,000,000 copies of Linux are now in use. Speaking of the alternate meaning of "free," about half of these copies are purchased from Linux vendors.

Linux largely owes its success to one killer application – the Apache Web Server, which is used on about 50% of all Internet sites.

Martin's book revolves around Linux as the prime example of open source.

### **History**

The concept of open source started in 1984 with the formation of the Free Software Foundation.

---

<sup>1</sup> Fink, M., *The Business and Economics of Linux and Open Source*, Prentice Hall PTR; 2003.

Linux was first released in 1991. It had, in fact, simply been a college project of Linus Torvalds, who wanted a version of UNIX to run on his Intel-based 386 PC. He released it as free software, and a multitude of other developers began porting it to other platforms and adding enhancements.

Today, Linux is the most widely ported of all operating systems. Most major hardware and software vendors support it, including HP, IBM, Dell, Oracle, SAP, and BEA.. It is strongly supported by a variety of tools, especially the GNU (GNU is Not Unix) compilers, editors, and other tools.

Red Hat, the first Linux distributor, was founded in 1994. Today, Linux is widely distributed by a variety of vendors, including Red Hat, SuSE, and many others,

The term “open source” was coined in 1998 as a more meaningful and less confusing term for “free software.”

## Open Source Development Model

An open source project is generally sponsored by a corporation, foundation, committee, or individual. The development of the project proceeds with the efforts of a worldwide, loosely coupled community of developers who are interested in the project.

There are a variety of sponsors for the multitude of open source projects, including;

- Linux and its components
- Web Services and Application Servers (Apache, JBOSS)
- Languages (GCC, Perl, Python)
- Desktop and Office Productivity (GNOME, Mozilla)
- Databases (MySQL)
- PDA (handheld Linux)
- Clusters (Beowulf)

For each sponsor, there is typically one individual who assumes the role of *maintainer*. It is the maintainer who accepts or rejects code for the project. In the case of Linux, however, the development effort is so large that there is a hierarchy of maintainers working under the direction of Linus Torvalds. Linus remains the maintainer for a large part of the Linux kernel.

It is the maintainer who decides when to issue a release. Interim releases will be issued quite frequently. They may be buggy, but the intent is to get a lot of eyes on the code to make it stable and complete. A production release will be issued when and only when the maintainer decides that the code is ready. He is not under any pressure of committed deadlines. The schedules are what they are.

Code quality is basically a matter of trust and respect within the development community. Code additions and modifications are viewed by many, and the community is quick to praise and to criticize. Developers who consistently deliver faulty code are not welcome in the community and are likely to have their code rejected by the maintainer.

This development culture is indeed quite different from the corporate development culture, and this is a fundamental reason why the concept of open source is so often difficult to sell in an organization. A classic describing the difference in philosophy between corporations and the open source community is the book, *Cathedral and the Bazaar: Musings on Linux and Open Source by*

*an Accidental Revolutionary*, by Eric Raymond.<sup>2</sup> The cathedral is the traditional software development model. The Bazaar is the open source development model.

## **Distributions**

There are many distributors of Linux. Red Hat and SuSE have already been mentioned. Other major open source projects have their own set of distributors working in a similar fashion to the Linux distributors.

A Linux distributor will combine the Linux kernel with compilers, editors, and tools from GNU as well as with other open source components. They may then add their own enhancements to meet a particular market need.

Distributors often will bundle their open source products with optional services and support. This is one business model to make money.

Because each distributor may add its own enhancements to the open source product, Linux was threatened with the UNIX syndrome of many incompatible versions. To avoid this pitfall, the Free Standards Group was established. Its role is to establish standards that all distributors will follow. It does this not by writing new standards but by documenting what exists today.

This group has completed a Linux standard called the Linux Standards Base (LSB).

## **Business Benefits**

As a radically new and different approach to software development and acquisition, open source comes with its own set of business benefits and concerns.

At the top of the benefits list is the total cost of ownership, or TCO, of an open source-based system. Low TCO starts with the low acquisition cost of an open source product. It can often be obtained at no cost by downloading it from the Internet. Alternatively, it can often be purchased from a distributor for a cost approximating the media and manual costs.

Open source products run on commodity hardware, thus greatly reducing hardware acquisition costs compared to proprietary systems.

Support can also be obtained at no cost if one is in a position to accept ad hoc help from the user community. Alternatively, since the source code is freely available, inhouse technical staff can provide support. Finally, support can be purchased from any one of a number of vendors who have built up a support business around the freely available source code. Since there are often a variety of vendors offering support, support costs can be competitively reasonable.

Because open source is now a well-supported topic in colleges and trade schools, there is a good availability of trained talent.

Open source products are vendor neutral since they have been developed by a wide-ranging community. Therefore, there is no commitment to proprietary hardware or software.

There is no formal end-of-life associated with an open source product. It does not become "functionally stabilized," nor is support terminated at some point in the future. It will always be supported so long as there is interest in the development community. Should support lag after a

---

<sup>2</sup> Available from Amazon.com or free on many Web sites. Just Google it.

time, the using organization always has the capability to take over its own support. This is typically not possible with proprietary software.

Finally, since an organization has access to the source code, it can customize the product for its own needs and remove its reliance on the disparate development community. As discussed later, this is no problem if the product is to be used internally. If it is to be resold, there are significant licensing requirements with which one must comply..

## **Business Concerns**

Notwithstanding the significant business benefits of open source, there are several serious concerns that an organization faces.

There is a concern that the open source development community may be unreliable. There is no formal accountability for product quality. There is, in fact, no control over the development effort – its product direction, its schedules, its plans, its road maps. Even worse, there typically are no plans or road maps.

The company cannot own the intellectual property that is included in open source projects of its employees.

Since the development responsibility is so spread out, it often is not possible to get indemnification that someone else's intellectual property is being used improperly. This applies even to acquiring an open source product from a distributor.

There is often no way to get a service level agreement for downtime and other malfunctions even if the open source product is purchased through a distributor.

There may be government regulatory issues. For instance, many countries prohibit the export of encryption products for security reasons. As a result, open source projects such as these are often carried out in countries with no such restrictions. The U.S. government has recently eased these restrictions by allowing the export of open source encryption so long as the relevant government agencies are notified.

## **Business Uses**

Given the pros and cons of open source, there are several useful ways in which open source can be put to work for the benefit of a company.

### ***Inhouse Use***

If an open source product is only going to be used internally, there are no restrictions. It may be freely modified and used in any way. Support can be purchased externally or provided by inhouse staff.

The only caveat is that a decision cannot be made later to distribute it without carefully evaluating the changes that have been made to the open source portion of the product. These changes will have to be made available to the community at large under an open source license.

## **Hardware Vendors**

It is common today for hardware vendors to take advantage of open source in two ways. They may make their products compatible with open source products, or they may create open source products to support their hardware.

In either case, it is valuable for them to participate with the development community to ensure that their open source facilities are kept up-to-date and provide the ever-expanding support that the hardware may require.

## **Software Vendors**

The most difficult decisions have to be made by those software vendors who currently market proprietary products. They could avoid open source all together but face a long-term product devaluation as open source products impinge on their territory.

Particularly, if they are not in a niche market but rather are in a market of wide interest, it is a virtual certainty that one or more open source products will begin to offer serious competition at some time in the future. The timing cannot be predicted; but when it does occur, the open source product can move to meet market demands very quickly and can make enhancements available at no charge.

This is a difficult movement to oppose. Software patents and defense of intellectual property rights have generally not been successful. It is simply too easy for the development community to rewrite its code so that it is not infringing. Besides, who are you going to sue? The end user? This has certainly backfired in the past.

Therefore, there should be strong motivation to join the open source movement rather than to fight it. But this does not mean abandoning the corporate business model for making money. A key strategy is to work high on the stack to offer proprietary products while cooperating with the open source community to contribute to the lower layers of the stack to keep it moving in the direction needed by the proprietary products. In this way, maximum benefit is achieved from the open source community in optimizing the lower layers while keeping the upper layers proprietary.

Especially for software vendors, there is a tremendous leap of faith to move from inhouse development to open source development. Many of the problems have been discussed above – a large, unmanageable team, frequent buggy releases, no planning or road maps, no guarantee that market requirements will be met.

On the other hand, as also discussed above, open source brings with it a number of significant advantages – free development, rapid response to changing market conditions, a plethora of highly qualified people, and an ultimate quality product. As Eric Raymond, author of *Cathedral and the Bazaar* said, “Given enough eyeballs, all bugs are shallow.”

Given the wide disparity between corporate culture and open source culture, Martin Fink has made an interesting and powerful attempt to contrast the two for the wavering manager in his Chapter 9, [The Corporate Bazaar](#). In this chapter, he overlays a standard corporate organization onto the open source model and relates each function quite succinctly.

## **Open Source Licensing**

The basic rule in open source licensing is reciprocity. If open source code is used in a product that is to be distributed or sold, that part that is the original open source code must be distributed

under an equivalent open source license. This applies even if the open source code has been modified. All of the modifications must be distributed under an equivalent license.

The one exception is that code that uses open source but does not integrate with it (for instance, it sits on top of an open source product and accesses it only through a public interface) can be sold under a proprietary corporate license.

There are many open source licenses. The most common is the GPL (GNU General Purpose License). All licenses must be certified by the Open Source Initiative in order to qualify as an open source license. These licenses generally require that

- all applicable source code is freely distributed.
- the source code is available to all licensees.
- the licensee is free to modify the code and to redistribute it.
- the author is known so that the integrity of the code is guaranteed.
- its use cannot be limited to a specific product.
- it must not discriminate against any user group or application.
- it cannot contaminate the licenses of other software distributed with it.
- all derivative works must be licensed to everyone under an equivalent license (known as copyleft).

As we have previously said, these license restrictions apply only to open source which is to be redistributed. If it is to be used only for internal purposes, there are no licensing requirements.

Dual licensing is allowed. That is, an open source product provided by a company (that is, the company is the developer, is the maintainer, and owns the copyright to the code) can be offered both under an open license and a commercial license. The advantage of this is that the open license serves to increase the user base and the awareness of the product in the marketplace. The commercial license can charge a fee such as a royalty and can be made attractive to the marketplace as an upgrade to the open source license. For instance, it could be better tested, it could contain special enhancements, new versions could be released earlier than the open source versions, and it could commit to scheduled roadmaps. No reciprocity is required under the commercial license.

Linux is available only under an open source license. It is not available under a commercial license.

## **Business Models**

The above discussions lead to the real conclusion of the book – how do you make money in open source. There are three basic models:

- Outbound open source – move proprietary products to open source.
- Inbound open source – integrate open source into your proprietary products.
- Internal use – bring in open source to benefit your own internal operations.

These lead to the following strategies:

- Create proprietary software (or hardware) that runs on top of open source and sell it.
- Provide support and services for your open source products.
- Use dual licensing to create a body of users of your open source products and to establish a credible record of quality so that users will turn to you for proprietary versions as well as support and services.

- Contribute to open source projects that are important to you to keep them moving in the right direction.

## **Summary**

Whether management likes it or not, open source software is gaining momentum. At some time in the future, most of us are going to have to deal with it. The question isn't "if." It is "when."

Martin Fink's book is an excellent tutorial on details of open source for those managers who want to dip a toe into this rising tide, who want to know more about it so that they can decide whether or not to embrace it, or who want to dig in for a fight. It is easy to read and is quite complete.

[Editor's Note: Though open source is generally associated with commodity hardware, it is important to note that many proprietary systems also support open source. Open source software runs on HP's NonStop servers under the OSS personality. Linux is supported on Stratus' ftServers and is being ported to IBM System 390 mainframes.]